

# Deber 03

Nicolas Romero

00212949

## CMP-4005 -- Homework 3

1.

a)

Primero obtenemos la ip de google haciendo ping a [google.com](http://google.com) para poder ver en Wireshark cuales son los paquetes que se mandan hacia esta direccion

```
Pinging google.com [2800:3f0:4005:409::200e] with 32 bytes of data:
Reply from 2800:3f0:4005:409::200e: time=14ms
Reply from 2800:3f0:4005:409::200e: time=14ms
Reply from 2800:3f0:4005:409::200e: time=14ms
Reply from 2800:3f0:4005:409::200e: time=14ms
```

Con esto analizamos y buscamos en Wireshark

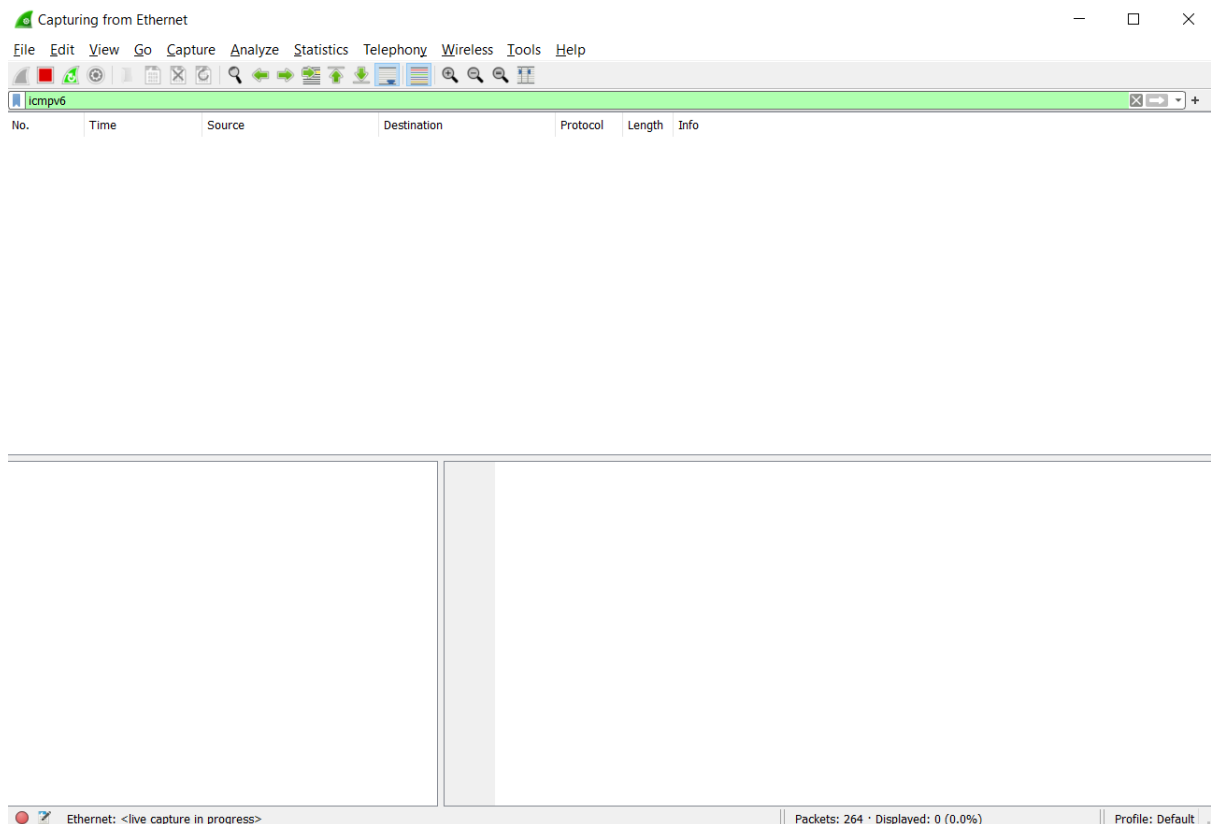
1429...	690.983593	198.251.234.236	192.168.100.181	UDP	69 8801 → 55957 Len=27
1429...	691.002872	198.251.234.236	192.168.100.181	UDP	288 8801 → 49924 Len=246
1429...	691.023400	198.251.234.236	192.168.100.181	UDP	294 8801 → 49924 Len=252
1429...	691.028478	2800:b0:144:dd1:9c...	2800:3f0:4005:409::...	ICMPv6	126 Echo (ping) request id=0x0001, seq=231, hop limit=3 (no respo...
1429...	691.033177	192.168.100.181	198.251.234.236	UDP	69 55957 → 8801 Len=27
1429...	691.042205	198.251.234.236	192.168.100.181	UDP	69 8801 → 55957 Len=27

Ejecutamos el tracert [google.com](http://google.com) para buscar estos paquetes

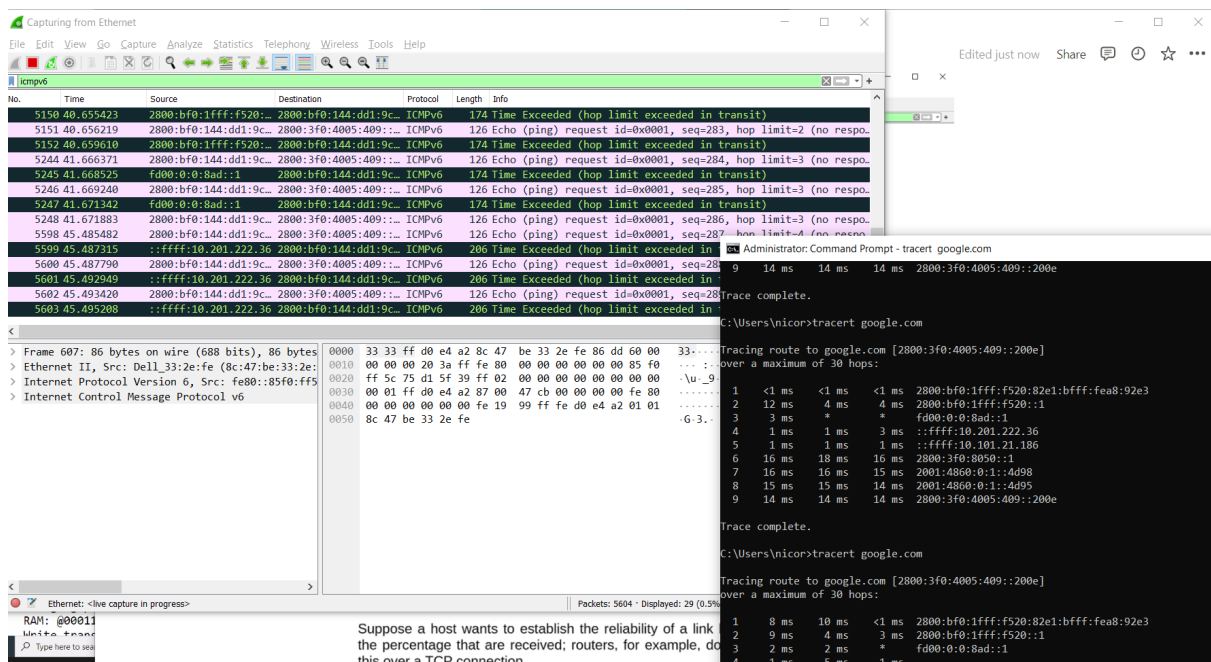
The screenshot shows a Wireshark capture of network traffic and a Windows Command Prompt window. The Wireshark capture shows several UDP and ICMPv6 packets. The Command Prompt window shows the output of a tracert command to google.com, displaying the route taken by the packets and the time taken for each hop.

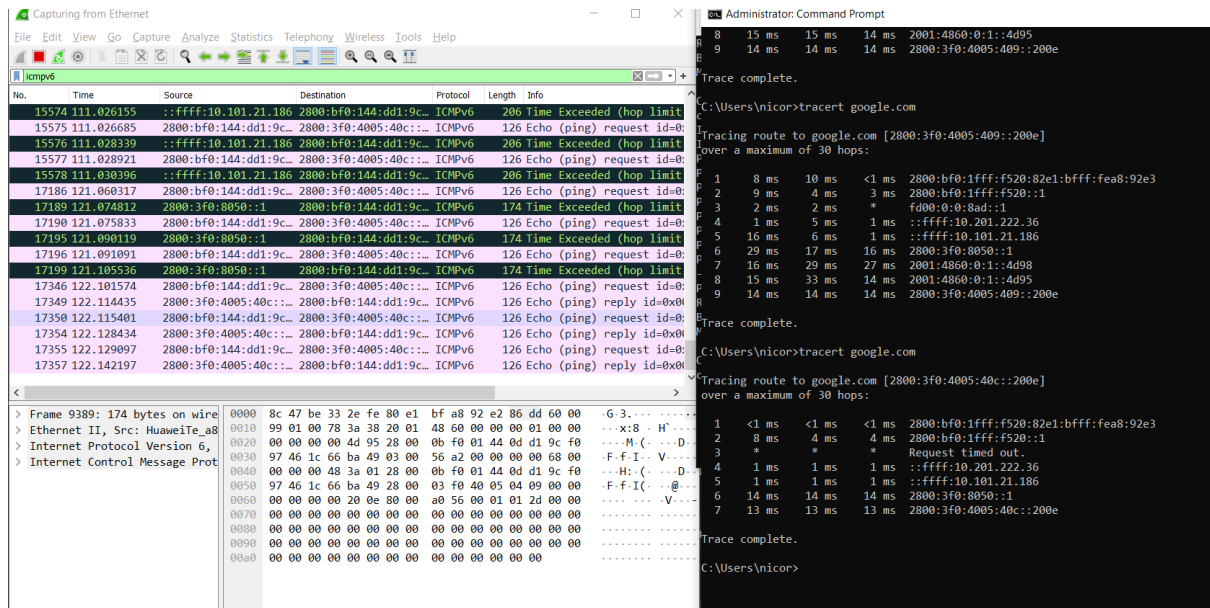
```
Tracing route to google.com [2800:3f0:4005:409::200e]
over a maximum of 30 hops:
  0  0 ms  0 ms  0 ms  2800:b0:144:dd1:9c::1
  1  <1 ms  <1 ms  <1 ms  2800:b0:1fff:f520:82e1:bfff:fea8:92e3
  2  12 ms  4 ms  4 ms  2800:b0:1fff:f520::1
  3  3 ms  *  *  fd00:0:0:8ad::1
  4  1 ms  1 ms  3 ms  :ffff:10.201.222.36
  5  1 ms  1 ms  1 ms  :ffff:10.101.21.186
  6  16 ms  18 ms  16 ms  2800:3f0:8050::1
  7  16 ms  16 ms  15 ms  2001:4860:0:1::4d98
  8  15 ms  15 ms  14 ms  2001:4860:0:1::4d95
  9  14 ms  14 ms  14 ms  2800:3f0:4005:409::200e
Trace complete.
```

viendo el paquete en Wireshark podemos saber que el protocolo que se usa es ICMPV6 por lo que usamos el filtro solo para capturar paquetes con este protocolo



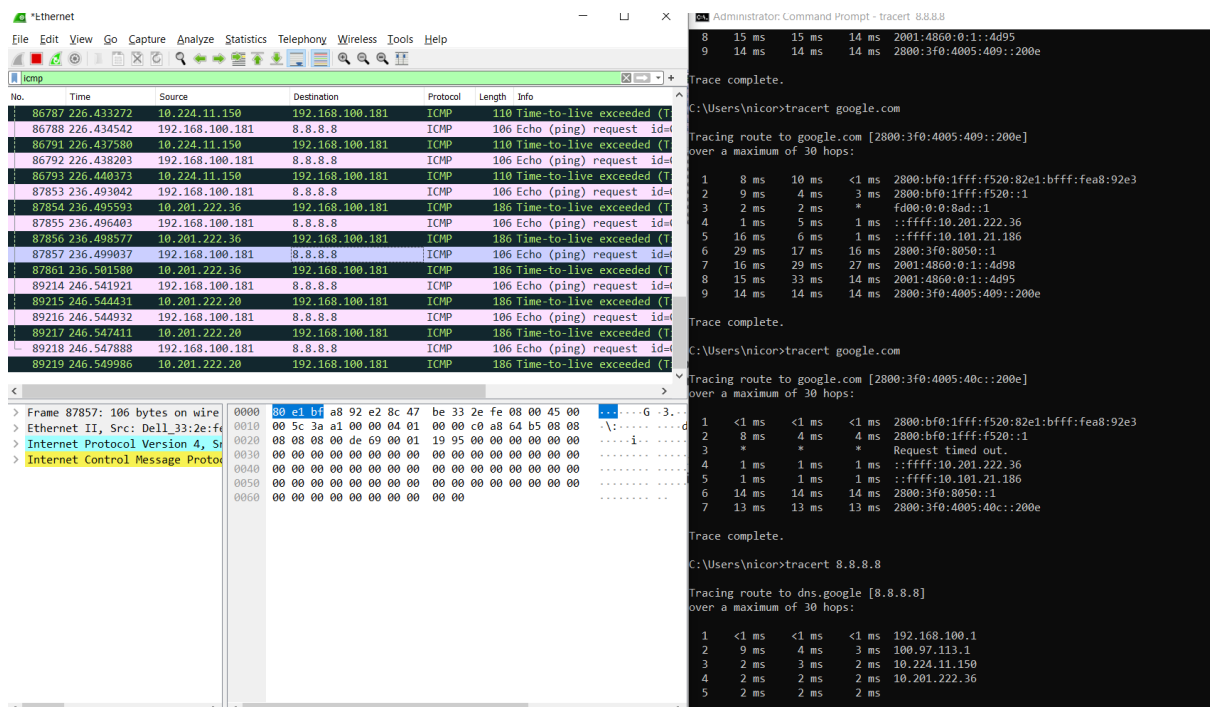
ya con el filtro, como podemos ver no captura nada todavia hasta que ejecutamos el tracert route a google.com





Realizamos este procesos repetidas veces [10 en total]

Tambien lo podemos hacer hacia la ip 8.8.8.8 = dns.google en este caso el paquete tiene un protocolo de ICMP



b)

Para capturar TCP Handshake lo que vamos hacer es ingresar a mi sitio web de mi App

sietetomantodos.tech

```
C:\Users\nicor>ping sietetomantodos.tech

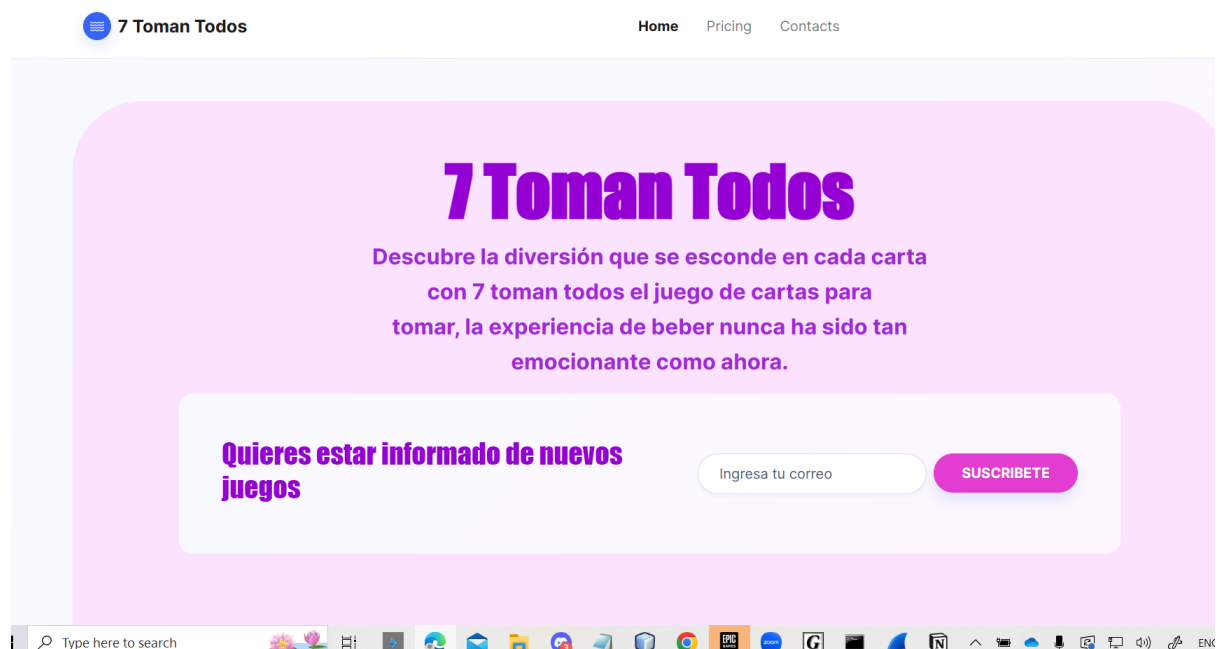
Pinging sietetomantodos.tech [137.184.90.109] with 32 bytes of data:
Reply from 137.184.90.109: bytes=32 time=122ms TTL=53
Reply from 137.184.90.109: bytes=32 time=121ms TTL=53
Reply from 137.184.90.109: bytes=32 time=121ms TTL=53
Reply from 137.184.90.109: bytes=32 time=121ms TTL=53

Ping statistics for 137.184.90.109:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 121ms, Maximum = 122ms, Average = 121ms

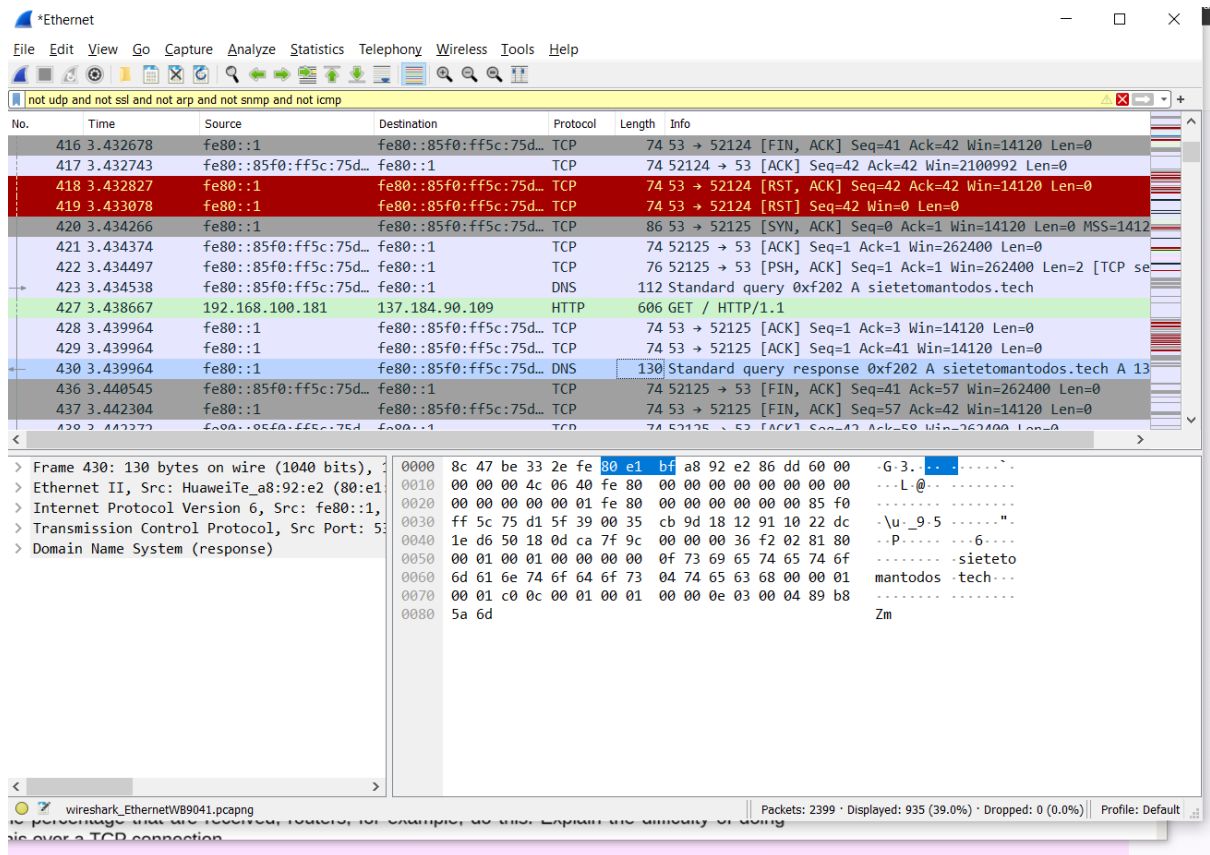
C:\Users\nicor>tracert sietetomantodos.tech
```

realizamos un ping para saber la ip, esto lo podria saber yo revisando donde se encuentra configurado el DNS y los archivos cargados en el servidor en este caso la pagina esta mantenida por DigitalOcean, la ip obtenida es 137.184.90.109

empezamos la captura con wireshark

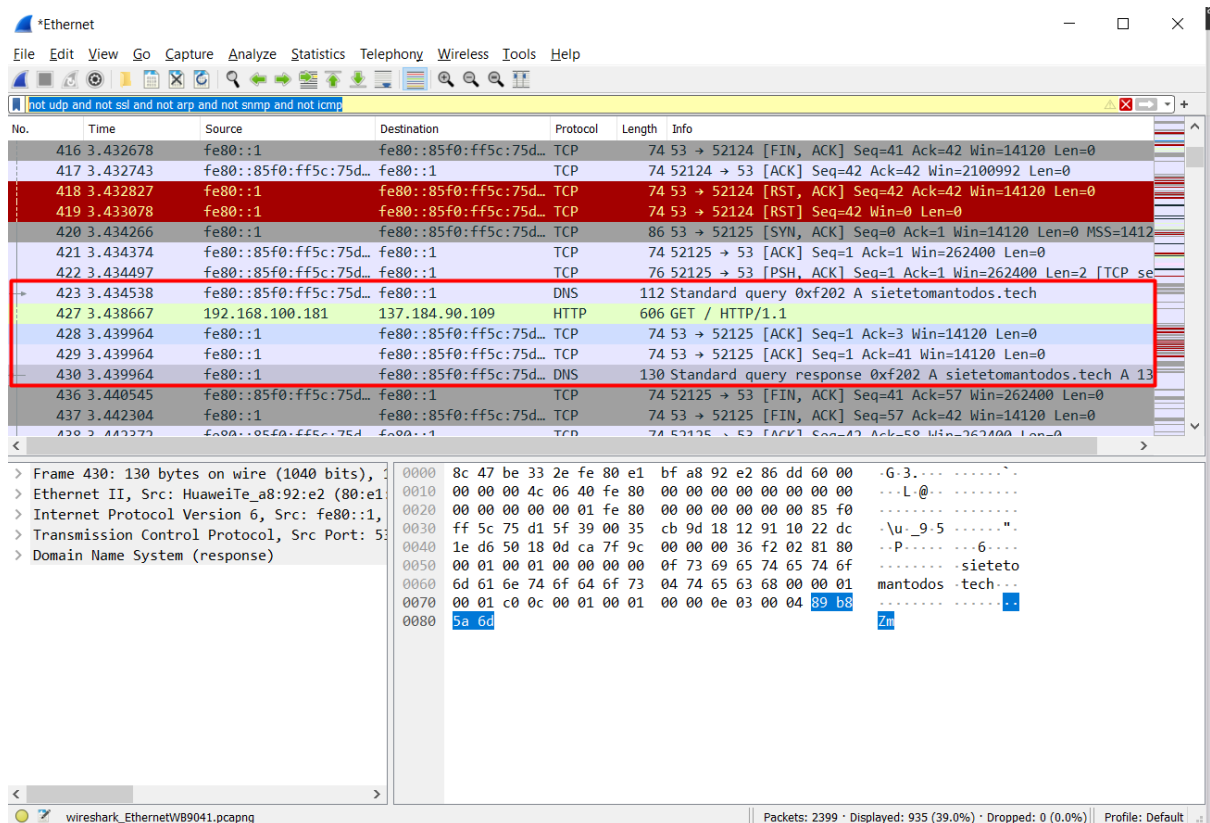


Entramos a la pagina web

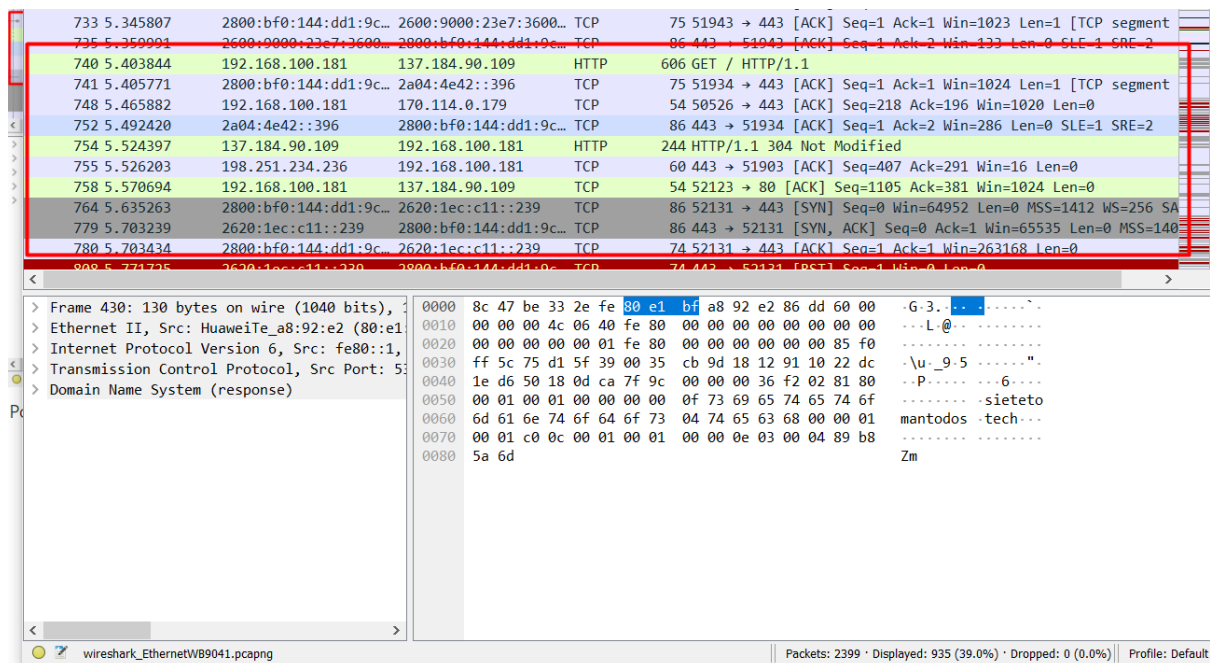


y dejamos de capturar, aplicamos el filtro “not udp and not ssl and not arp and not snmp and not icmp”

para que solo se nos muestre los paquetes con protocolo TCP



Podemos ver que se ingreso a la pagina y se capturo el handshake



mas abajo si volvemos a ingresar y capturamos podemos ver nuestro [SYN] sincronize con el servidor y el servidor responde con un [ACK] acknowledgment



File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp

No.	Time	Source	Destination	Protocol	Length	Info
3385	8.703866	2800:bf0:144:dd1:9c...	2a04:4e42:8a::272	TCP	86	52928 → 443 [ACK] Seq=177 Ack=98145 Win=102
3389	8.731762	2800:2a0:11:82:b52...	2800:bf0:144:dd1:9c...	TCP	74	443 → 53488 [ACK] Seq=25781 Ack=13436 Win=5
3390	8.731956	192.168.100.181	52.94.236.248	TCP	54	52897 → 443 [FIN, ACK] Seq=1 Ack=1 Win=102
3391	8.732365	192.168.100.181	192.168.100.1	TCP	66	53564 → 53 [SYN] Seq=0 Win=64240 Len=0 MSS=
3392	8.732560	192.168.100.181	192.168.100.1	TCP	66	53565 → 53 [SYN] Seq=0 Win=64240 Len=0 MSS=
3393	8.732740	192.168.100.181	192.168.100.1	TCP	66	53566 → 53 [SYN] Seq=0 Win=64240 Len=0 MSS=
3394	8.732793	192.168.100.1	192.168.100.181	TCP	66	53 → 53564 [SYN, ACK] Seq=0 Ack=1 Win=14606
3395	8.732936	192.168.100.181	192.168.100.1	TCP	54	53564 → 53 [ACK] Seq=1 Ack=1 Win=2102272 Le
3396	8.732941	192.168.100.1	192.168.100.181	TCP	66	53 → 53565 [SYN, ACK] Seq=0 Ack=1 Win=14606
3397	8.732980	192.168.100.181	192.168.100.1	TCP	56	53564 → 53 [PSH, ACK] Seq=1 Ack=1 Win=21022
3398	8.733017	192.168.100.181	192.168.100.1	DNS	97	Standard query 0x042b AAAA webshell.suite.c
3399	8.733056	192.168.100.1	192.168.100.181	TCP	66	53 → 53566 [SYN, ACK] Seq=0 Ack=1 Win=14606
3400	8.733078	192.168.100.181	192.168.100.1	TCP	54	53565 → 53 [ACK] Seq=1 Ack=1 Win=262656 Le
3401	8.733109	192.168.100.181	192.168.100.1	TCP	56	53565 → 53 [PSH, ACK] Seq=1 Ack=1 Win=26265
3402	8.733122	192.168.100.181	192.168.100.1	TCP	54	53566 → 53 [ACK] Seq=1 Ack=1 Win=2102272 Le

> Frame 5: 66 bytes on wire (528 bits),  
 > Ethernet II, Src: HuaweiTe\_a8:92:e2 (8  
 > Internet Protocol Version 4, Src: 66.1  
 > Transmission Control Protocol, Src Port

0000 8c 47 be 33 2e fe 80 e1 bf a8 92 e2 08 00 45 00 -G-3... ..E-  
 0010 00 34 00 00 40 00 31 06 b0 b6 42 6e 31 42 c0 a8 -4-@1-..Bn18..  
 0020 64 b5 01 bb d1 0a dc d8 9a 43 04 0a bb a3 80 12 d-...f-..C-.....  
 0030 ff ff cd 95 00 00 02 04 05 84 01 03 06 04 02 ....E-.....  
 0040 00 00 ..

Aqui podemos ver otro ejemplo ingresado a [amazon.com](https://www.amazon.com) se hace un [SYN] con el servidor, luego tenemos un [SYN,ACK] y luego se devuelve un [ACK] informando que se recibio el mensaje, por lo que esta seria la captura TCP handshake

\*ethernet

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter --> <Ctrl-F>

No.	Time	Source	Destination	Protocol	Length	Info
885	8.943352	198.251.234.236	192.168.100.181	UDP	297	8801 → 49924 Len=255
886	8.944319	192.168.100.181	198.251.234.236	TCP	54	51903 → 443 [ACK] Seq=1572 Ack=1105 Win=1025 Len=0
887	8.948369	146.75.106.132	192.168.100.181	TLSv1.2	105	Application Data
888	8.948369	146.75.106.132	192.168.100.181	TLSv1.2	78	Application Data
889	8.948535	192.168.100.181	146.75.106.132	TCP	54	52882 → 443 [ACK] Seq=1 Ack=76 Win=1023 Len=0
890	8.948585	192.168.100.181	146.75.106.132	TCP	54	52882 → 443 [FIN, ACK] Seq=1 Ack=76 Win=1023 Len=0
891	8.948629	146.75.106.132	192.168.100.181	TCP	60	443 → 52882 [FIN, ACK] Seq=76 Ack=1 Win=290 Len=0
892	8.948678	192.168.100.181	146.75.106.132	TCP	54	52882 → 443 [ACK] Seq=2 Ack=77 Win=1023 Len=0
893	8.964043	198.251.234.236	192.168.100.181	UDP	283	8801 → 49924 Len=241
894	8.984662	198.251.234.236	192.168.100.181	UDP	281	8801 → 49924 Len=239
895	8.992845	2a03:2880:f22b:c4:f... 2800:bf0:144:dd1:9c...	2800:bf0:144:dd1:9c...	TLSv1.2	102	Application Data
896	8.994153	2a03:2880:f22b:c4:f... 2800:bf0:144:dd1:9c...	2800:bf0:144:dd1:9c...	TLSv1.2	102	Application Data
897	8.994982	198.251.234.236	192.168.100.181	UDP	287	8801 → 49924 Len=245
898	8.015740	198.251.234.236	192.168.100.181	UDP	292	8801 → 49924 Len=250
899	8.020847	198.251.234.236	192.168.100.181	TCP	60	443 → 51903 [ACK] Seq=1105 Ack=1572 Win=16 Len=0
900	8.034395	2800:bf0:144:dd1:9c... 2a03:2880:f22b:c4:f...	2a03:2880:f22b:c4:f...	TCP	74	50577 → 443 [ACK] Seq=33 Ack=29 Win=1026 Len=0
901	8.034395	2800:bf0:144:dd1:9c... 2a03:2880:f22b:c4:f...	2a03:2880:f22b:c4:f...	TCP	74	60697 → 443 [ACK] Seq=33 Ack=29 Win=1025 Len=0
902	8.035751	146.75.106.132	192.168.100.181	TCP	60	443 → 52882 [ACK] Seq=77 Ack=2 Win=290 Len=0
903	8.057126	198.251.234.236	192.168.100.181	UDP	306	8801 → 49924 Len=264
904	8.057126	198.251.234.236	192.168.100.181	UDP	312	8801 → 49924 Len=270
905	8.077732	198.251.234.236	192.168.100.181	UDP	310	8801 → 49924 Len=268
906	8.077732	192.168.100.181	198.251.234.236	UDP	117	49924 → 8801 Len=75
907	8.119249	198.251.234.236	192.168.100.181	UDP	304	8801 → 49924 Len=262
908	8.119249	198.251.234.236	192.168.100.181	UDP	302	8801 → 49924 Len=260
909	8.139727	198.251.234.236	192.168.100.181	UDP	308	8801 → 49924 Len=266
910	8.160518	198.251.234.236	192.168.100.181	UDP	304	8801 → 49924 Len=262
911	8.181118	198.251.234.236	192.168.100.181	UDP	309	8801 → 49924 Len=267
912	8.201778	198.251.234.236	192.168.100.181	UDP	311	8801 → 49924 Len=269
913	8.222206	198.251.234.236	192.168.100.181	UDP	313	8801 → 49924 Len=271
914	8.231390	192.168.100.181	52.226.139.180	TLSv1.2	97	Application Data
915	8.242904	198.251.234.236	192.168.100.181	UDP	309	8801 → 49924 Len=267
916	8.262139	192.168.100.181	198.251.234.236	UDP	123	49924 → 8801 Len=81
917	8.263633	198.251.234.236	192.168.100.181	UDP	294	8801 → 49924 Len=252
918	8.284341	198.251.234.236	192.168.100.181	UDP	288	8801 → 49924 Len=256

> Frame 5: 66 bytes on wire (528 bits), 66 bytes captured (528 b  
 > Ethernet II, Src: HuaweiTe\_a8:92:e2 (80:e1:bf:a8:92:e2), Dst: 1  
 > Internet Protocol Version 4, Src: 66.110.49.66, Dst: 192.168.1  
 > Transmission Control Protocol, Src Port: 53, Dst Port: 443

0000 8c 47 be 33 2e fe 80 e1 bf a8 92 e2 08 00 45 00 -G-3... ..E-  
 0010 00 34 00 00 40 00 31 06 b0 b6 42 6e 31 42 c0 a8 -4-@1-..Bn18..  
 0020 64 b5 01 bb d2 66 c0 16 35 5f 50 61 c8 e6 80 12 d-...f-..5\_Pa....  
 0030 ff ff 45 00 00 02 04 05 84 01 03 06 04 02 ....E-.....

wireshark\_EthernetA9H341.pcapng Packets: 918 · Displayed: 918 (100.0%) Profile: Default

Esta es la pagina de congestion sin aplicar los filtros

Ethernet

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.flags.syn==1 && tcp.flags.ack==1

No.	Time	Source	Destination	Protocol	Length	Info
85	1.850357	192.168.100.1	192.168.100.181	TCP	66	53 → 55752 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
89	1.851123	192.168.100.1	192.168.100.181	TCP	66	53 → 55754 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
98	1.851870	192.168.100.1	192.168.100.181	TCP	66	53 → 55753 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
112	1.858340	192.168.100.1	192.168.100.181	TCP	66	53 → 55756 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
113	1.858340	192.168.100.1	192.168.100.181	TCP	66	53 → 55758 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
116	1.858601	192.168.100.1	192.168.100.181	TCP	66	53 → 55755 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
117	1.858601	192.168.100.1	192.168.100.181	TCP	66	53 → 55757 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
119	1.858601	192.168.100.1	192.168.100.181	TCP	66	53 → 55759 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
120	1.858691	192.168.100.1	192.168.100.181	TCP	66	53 → 55760 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
132	1.859573	192.168.100.1	192.168.100.181	TCP	66	53 → 55761 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
145	1.860017	192.168.100.1	192.168.100.181	TCP	66	53 → 55762 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
200	1.936523	192.168.100.1	192.168.100.181	TCP	66	53 → 55767 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
211	1.936730	192.168.100.1	192.168.100.181	TCP	66	53 → 55768 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
219	1.937766	192.168.100.1	192.168.100.181	TCP	66	53 → 55769 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
229	1.940229	192.168.100.1	192.168.100.181	TCP	66	53 → 55770 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
232	1.940606	192.168.100.1	192.168.100.181	TCP	66	53 → 55772 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
237	1.941131	192.168.100.1	192.168.100.181	TCP	66	53 → 55774 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
246	1.941498	192.168.100.1	192.168.100.181	TCP	66	53 → 55771 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
247	1.941498	192.168.100.1	192.168.100.181	TCP	66	53 → 55773 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
248	1.941498	192.168.100.1	192.168.100.181	TCP	66	53 → 55775 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
268	1.948303	192.168.100.1	192.168.100.181	TCP	66	53 → 55776 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
271	1.948546	192.168.100.1	192.168.100.181	TCP	66	53 → 55778 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
277	1.949162	192.168.100.1	192.168.100.181	TCP	66	53 → 55777 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
285	1.954630	192.168.100.1	192.168.100.181	TCP	66	53 → 55779 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
289	1.954963	192.168.100.1	192.168.100.181	TCP	66	53 → 55781 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
297	1.956922	192.168.100.1	192.168.100.181	TCP	66	53 → 55780 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
322	1.996399	204.79.197.203	192.168.100.181	TCP	66	443 → 55765 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 WS=256 SACK_PERM WS=128
341	2.018670	2880:2a0:11:82::b52_	2880:b0:144:dd1:9c_	TCP	86	443 → 55783 [SYN, ACK] Seq=0 Ack=1 Win=64800 Len=0 MSS=1440 SACK_PERM WS=128
343	2.019162	20.110.205.119	192.168.100.181	TCP	66	443 → 55766 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 WS=256 SACK_PERM WS=128
346	2.020523	2880:2a0:11:82::b52_	2880:b0:144:dd1:9c_	TCP	86	443 → 55784 [SYN, ACK] Seq=0 Ack=1 Win=64800 Len=0 MSS=1440 SACK_PERM WS=128
373	2.037850	104.86.191.35	192.168.100.181	TCP	66	443 → 55782 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1412 SACK_PERM WS=128
380	2.045330	20.189.173.5	192.168.100.181	TCP	66	443 → 55764 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 WS=256 SACK_PERM WS=128
398	2.115698	204.79.197.219	192.168.100.181	TCP	66	443 → 55786 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 WS=256 SACK_PERM WS=128
401	2.110262	204.79.197.219	192.168.100.181	TCP	66	443 → 55785 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 WS=256 SACK_PERM WS=128
496	2.223376	192.168.100.1	192.168.100.181	TCP	66	53 → 55790 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
502	2.227729	192.168.100.1	192.168.100.181	TCP	66	53 → 55789 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
503	2.227729	192.168.100.1	192.168.100.181	TCP	66	53 → 55791 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM WS=4
528	2.259964	104.86.191.35	192.168.100.181	TCP	66	443 → 55787 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1412 SACK_PERM WS=128

wireshark\_EthernetBLW941.pcapng

Packets: 6480 · Displayed: 271 (4.2%) · Dropped: 0 (0.0%)

Profile: Default

Aqui la congestion solo de TCP handshake

2.



# Pregunta (5)

Nodo A

A	D	2
A	E	7
A	B	4
A	C	7

Nodo B

B	A	4
B	C	3
B	D	2
B	E	2

Nodo E

E	A	6
E	B	2
E	C	1
E	D	4

## Tabla por Saltos (Routing Table)

Nodo A

Nodo destino	Hop	Cost
D	D	2
E	D	7
B	D	4
C	D → B → E	7

Nodo B

Nodo Destino	Hop	Cost
A	D	4
C	E	3
D	D	2
E	E	2

Nodo E

Nodo Destino	Hop	Cost
A	B → D	6
B	B	2
C	C	1
D	B	4

3.

Suppose a host wants to establish the reliability of a link by sending packets and measuring the percentage that are received; routers, for example, do this. Explain the difficulty of doing this over a TCP connection.

La retransmisión de paquetes perdidos y el control de flujo son sólo un par de los métodos fundamentales que incorpora TCP para garantizar la fiabilidad de la conexión. Dado que TCP intentará recuperar automáticamente los paquetes perdidos sin indicar claramente la pérdida en la capa del router, estos métodos internos pueden ocultar la pérdida genuina de paquetes en la capa del router.

por lo que a la falta de acceso directo a los paquetes en los routers, la posibilidad de pérdida en los enlaces, los mecanismos internos de fiabilidad de TCP y otros factores, medir la fiabilidad de un enlace a través de una conexión TCP se vuelve complicado.

4.

Consider a simple congestion control algorithm that uses linear increase and multiplicative decrease (no slow start). Assume the congestion window size is in units of packets rather than bytes, and it is one packet initially.

a) Give a detailed sketch of this algorithm.

se va a ir aumentando el tamaño de la ventana de congestion de forma gradual

b)-c)

# Pregunta ④

b)  $\frac{RTT}{Sender} = \begin{array}{|c|c|c|c|} \hline 1 & 2 & 3 & 4 \\ \hline 1 & 2+3 & 4+6 & 7+10 \\ \hline \end{array} \quad \begin{array}{l} \text{Packet 9} \\ \text{lost} \end{array}$

$\frac{RTT}{Sender} = \begin{array}{|c|c|c|c|c|} \hline 5 & 6 & 7 & 8 & 9 \\ \hline 9+10 & 11+13 & 14+17 & 18+22 & 23+28 \\ \hline \end{array} \quad \begin{array}{l} \text{Packet 25} \\ \text{lost} \end{array}$

$\frac{RTT}{Sender} = \begin{array}{|c|c|} \hline 10 & 11 \\ \hline 25+27 & 28+31 \\ \hline \end{array} \quad \begin{array}{l} \text{Packet 30} \\ \text{lost} \end{array}$

$\frac{RTT}{Sender} = \begin{array}{|c|c|c|} \hline 12 & 13 & 14 \\ \hline 30+31 & 32+34 & 35+38 \\ \hline \end{array} \quad \begin{array}{l} \text{Packet 38} \\ \text{lost} \end{array}$

$\frac{RTT}{Sender} = \begin{array}{|c|c|c|c|} \hline 15 & 16 & 17 & 18 \\ \hline 38+39 & 40+42 & 43+46 & 47+50 \\ \hline \end{array} \quad \begin{array}{l} \text{Packet 50} \\ \text{lost} \end{array}$

$\frac{RTT}{Sender} = \begin{array}{|c|} \hline 19 \\ \hline 50 \\ \hline \end{array} \quad \begin{array}{l} \text{Last package to} \\ \text{Send} \end{array}$

