



UNIVERSITAT<sup>DE</sup>  
BARCELONA

---

## Pràctica 5. El model OSI

---

Noah Márquez & Jan Morales

8 gener 2024

## ÍNDICE

1	Objectius de la pràctica	3
2	Exercici 1	3
3	Exercici 2	5
4	Exercici 3	10
5	Exercici 4	12
6	Conclusions	18

## 1. OBJECTIUS DE LA PRÀCTICA

L'objectiu principal de la pràctica és veure com s'encapsulen les diferents Unitats de Protocol d'Usuari (DPU) i que permeten transmetre informació entre dos equips de manera estàndard, independentment de les característiques dels equips en qüestió. Per assolir aquest objectiu utilitzarem un dels programes *sniffers* més utilitzats actualment: *Wireshark*

## 2. EXERCICI 1

*Wireshark* és un analitzador de protocols utilitzat per analitzar el trànsit de les xarxes de comunicació. Introduïm al lloc del filtre: **(ip.addr == 192.168.1.134) and (tcp)**. IP és l'adreça IP del nostre ordinador, i TCP és el protocol que escollim. Després de filtrar, podem mostrar la següent pantalla a *Wireshark*:

No.	Time	Source	Destination	Protocol	Length	Info
19703	694.024781	1.189.88.217	192.168.1.134	TCP	60	26881 → 63877 [RS
19704	694.550277	192.168.1.134	1.189.88.217	TCP	66	[TCP Retransmissi
19705	694.688335	192.168.1.134	154.160.5.50	TCP	66	[TCP Retransmissi
19706	694.688335	192.168.1.134	111.15.102.138	TCP	66	[TCP Retransmissi
19707	694.688360	192.168.1.134	42.233.131.212	TCP	66	[TCP Retransmissi
19708	694.688360	192.168.1.134	120.224.77.175	TCP	66	[TCP Retransmissi
19709	694.688400	192.168.1.134	117.182.145.236	TCP	66	[TCP Retransmissi
19710	694.688529	192.168.1.134	112.0.36.206	TCP	66	[TCP Retransmissi
19711	694.688614	192.168.1.134	112.42.8.12	TCP	66	[TCP Retransmissi
19712	694.888178	1.189.88.217	192.168.1.134	TCP	60	26881 → 63877 [RS
19713	695.392307	192.168.1.134	1.189.88.217	TCP	66	[TCP Retransmissi
19714	695.600842	192.168.1.134	41.138.91.190	TCP	55	[TCP ZeroWindowPr
19715	695.728512	1.189.88.217	192.168.1.134	TCP	60	26881 → 63877 [RS
19716	695.775778	192.168.1.134	222.134.91.202	TCP	54	63812 → 26881 [RS
19717	696.002987	192.168.1.134	183.250.243.150	TCP	126	[TCP Retransmissi
19718	696.235903	192.168.1.134	1.189.88.217	TCP	66	[TCP Retransmissi
19719	696.384441	192.168.1.134	222.186.57.113	TCP	126	[TCP Retransmissi

Figura 2.1: *Wireshark TCP/IP filter*

Des de la figura 2.1, podem veure clarament el trànsit de totes les xarxes de comunicació i la seva informació relacionada, on només mostrem únicament aquells paquets en els quals se segueixi el protocol TCP i es vegi implicada la nostra adreça IP, ja sigui com a adreça origen o adreça destí.

```
> Frame 19717: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits) on interface \Device
  ▾ Ethernet II, Src: IntelCor_9b:f5:eb (e4:5e:37:9b:f5:eb), Dst: zte_c0:19:de (24:d3:f2:c0:19:de)
    > Destination: zte_c0:19:de (24:d3:f2:c0:19:de)
    > Source: IntelCor_9b:f5:eb (e4:5e:37:9b:f5:eb)
    Type: IPv4 (0x0800)
    > Internet Protocol Version 4, Src: 192.168.1.134, Dst: 183.250.243.150
    > Transmission Control Protocol, Src Port: 63821, Dst Port: 26881, Seq: 1, Ack: 1, Len: 72
```

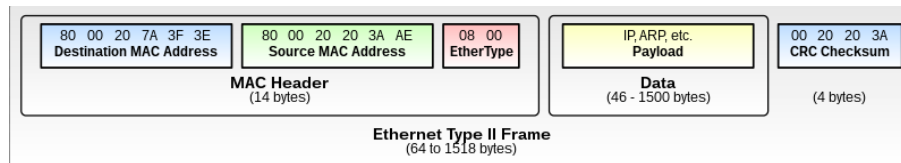
Figura 2.2: *Adreça MAC*

Seleccions aleatòriament un flux de xarxa i després seleccionem la capa Ethernet podem veure que hi ha dues adreces MAC, una és la **destinació** i l'altre és l'**origen**.

Podem entendre els components del protocol Ethernet mitjançant la trama Ethernet de tipus II a la figura 2.3. En primer lloc, la nostra Ethernet utilitza el protocol **Ethernet tipus II**. Al que hem de prestar atenció és a la part de la capçalera del MAC. Hi ha 6 bytes per mostrar l'adreça MAC de destinació, 6 bytes són l'adreça MAC d'origen i els 2 bytes restants són *EtherType*. El que s'utilitza és IPv4. La resta de dades, el CRC i el *checksum* formen part d'altres capes.

Podem comprovar que la direcció MAC té dos bits remarcats, les funcions dels quals són les següents:

- **LG Bit:** Indica si la direcció MAC ha estat assignada per un proveïdor o administrativa-ment.
- **IG Bit:** Indica si la direcció MAC és individual o de grup, valent 0 si és d'unidifusió o valent 1 si és de multidifusió o broadcast.

Figura 2.3: *Ethernet Type II Frame*

Finalment, podem veure la nostra adreça IP i la nostra adreça MAC a la part de Wi-Fi a través de la comanda **ipconfig -all**, que és la mateixa que l'adreça MAC d'origen a la Figura 2, i el protocol IPv4 es mostra en *EtherType* de *Ethernet Type II Frame*.

```

Adaptador de LAN inalámbrica Wi-Fi:

Sufijo DNS específico para la conexión. . . : Home
Descripción . . . . . : Intel(R) Wi-Fi 6 AX201 160MHz
Dirección física. . . . . : E4-5E-37-9B-F5-EB
DHCP habilitado . . . . . : sí
Configuración automática habilitada . . . : sí
Vínculo: dirección IPv6 local. . . : fe80::c52c:c0f5:19f6:284a%17(Preferido)
Dirección IPv4. . . . . : 192.168.1.134(Preferido)
Máscara de subred . . . . . : 255.255.255.0
Puerta de enlace predeterminada . . . . : 192.168.1.1
Servidor DHCP . . . . . : 192.168.1.1
IAID DHCPv6 . . . . . : 149184055
DUID de cliente DHCPv6. . . . . : 00-01-00-01-2A-AF-9C-60-D0-5F-64-3F-74-A5
Servidores DNS. . . . . : 212.230.135.2
                        212.230.135.1
NetBIOS sobre TCP/IP. . . . . : habilitado

```

Figura 2.4: *ipconfig -all*

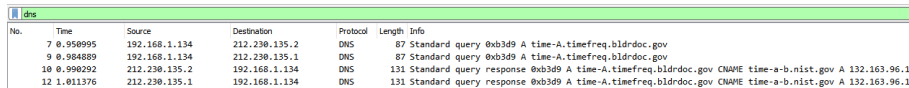
El programa també ens ofereix informació de com està formada la estructura de la nostra capçalera de direcció IP.

- **Version:** Ens indica si segueix el protocol IPv4 o IPv6, aquest cas com ens diu en nombre binari 01002 es la v4.
- **Header length:** Longitud de la capçalera.
- **Differentiated Services Field:** Informació de la qualitat del servei de comunicació.
  - **Differentiated Services Codepoint:** Diferenciar la qualitat de transmissió que tenen les dades que es transmeten.
  - **Explicit Congestion Notification:** Notificació de la congestió entre endpoints.
- **Total length:** Mida total del datagrama.
- **Identification:** Identificador únic del datagrama.
- **Flags:** Indicar si el paquet es pot fragmentar o no.
  - **Reserved Bit:** Sempre 0.
  - **Don't fragment**
  - **More fragments**
- **Fragment offset:** Posició del fragment dins del paquet.
- **Time to live:** Nombre de nodes pels quals pot ser enviat abans de ser descartat.

- **Protocol**
- **Header checksum:** detecció d'errors.
- **Source address**
- **Destination address**

### 3. EXERCICI 2

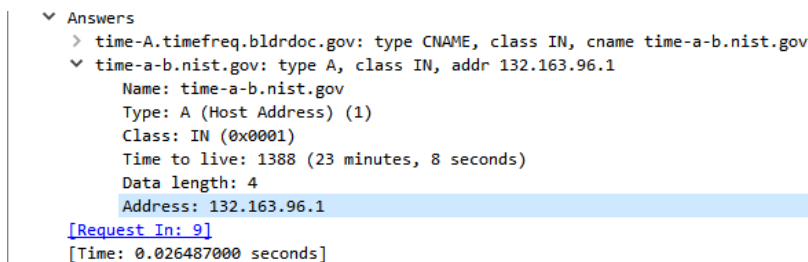
Quan accedim a aquest servidor mitjançant la comanda **telnet time-A.timefreq.blrdoc.gov13**, el *Wireshark* ens ajuda a capturar la següent informació:



No.	Time	Source	Destination	Protocol	Length	Info
7	0.958995	192.168.1.134	212.230.135.2	DNS	87	Standard query 0xb3d9 A time-A.timefreq.blrdoc.gov
9	0.964889	192.168.1.134	212.230.135.1	DNS	87	Standard query 0xb3d9 A time-A.timefreq.blrdoc.gov
10	0.998292	212.230.135.2	192.168.1.134	DNS	131	Standard query response 0xb3d9 A time-A.timefreq.blrdoc.gov CNAME time-a-b.nist.gov A 132.163.96.1
12	1.011376	212.230.135.1	192.168.1.134	DNS	131	Standard query response 0xb3d9 A time-A.timefreq.blrdoc.gov CNAME time-a-b.nist.gov A 132.163.96.1

Figura 3.1: *Wireshark* captura la informació

Al fer una sol·licitud al servidor, hem capturat que hem fet una sol·licitud a dos servidors DNS: **212.230.135.2** i **212.230.135.1**, tal com es mostra a la captura de pantalla de la figura 2.4. I el servidor DNS també ens ha enviat resposta i porta l'adreça IP del servidor al qual volem accedir: **132.163.96.1**.



```

Answers
  > time-A.timefreq.blrdoc.gov: type CNAME, class IN, cname time-a-b.nist.gov
  > time-a-b.nist.gov: type A, class IN, addr 132.163.96.1
    Name: time-a-b.nist.gov
    Type: A (Host Address) (1)
    Class: IN (0x0001)
    Time to live: 1388 (23 minutes, 8 seconds)
    Data length: 4
    Address: 132.163.96.1
  [Request In: 9]
  [Time: 0.026487000 seconds]
  
```

Figura 3.2: *DNS Response*

A la figura 3.2, podem veure clarament que la resposta que ens envia DNS va acompanyat de l'adreça IP del servidor al qual volem accedir.

Un cop determinada la IP, hem començat a analitzar els ports que es fan servir. Quan volem accedir al servidor DNS, utilitzem el protocol **UDP** i utilitzem el *port source 64042* per accedir, i el servidor DNS utilitza el *port 53* per acceptar la nostra sol·licitud. Al contrari, quan el servidor DNS ens vol enviar una resposta, utilitzen el port 53 per enviar el missatge, i nosaltres utilitzem el port 64042 per rebre el missatge:

No.	Time	Source	Destination	Protocol	Length	Info
7	0.950995	192.168.1.134	212.230.135.2	DNS	87	Standard query 0xt
9	0.984889	192.168.1.134	212.230.135.1	DNS	87	Standard query 0xt
10	0.990292	212.230.135.2	192.168.1.134	DNS	131	Standard query res
12	1.011376	212.230.135.1	192.168.1.134	DNS	131	Standard query res

> Frame 7: 87 bytes on wire (696 bits), 87 bytes captured (696 bits) on interface \Device\NPF\_{C...}  
 > Ethernet II, Src: IntelCor\_9b:f5:eb (e4:5e:37:9b:f5:eb), Dst: zte\_c0:19:de (24:d3:f2:c0:19:de)  
 > Internet Protocol Version 4, Src: 192.168.1.134, Dst: 212.230.135.2  
 > User Datagram Protocol, Src Port: 64042, Dst Port: 53  
 > Domain Name System (query)

Figura 3.3: Port request a DNS

No.	Time	Source	Destination	Protocol	Length	Info
7	0.950995	192.168.1.134	212.230.135.2	DNS	87	Standard query 0xt
9	0.984889	192.168.1.134	212.230.135.1	DNS	87	Standard query 0xt
10	0.990292	212.230.135.2	192.168.1.134	DNS	131	Standard query re
12	1.011376	212.230.135.1	192.168.1.134	DNS	131	Standard query re

> Frame 10: 131 bytes on wire (1048 bits), 131 bytes captured (1048 bits) on interface \Device\N...  
 > Ethernet II, Src: zte\_c0:19:de (24:d3:f2:c0:19:de), Dst: IntelCor\_9b:f5:eb (e4:5e:37:9b:f5:eb)  
 > Internet Protocol Version 4, Src: 212.230.135.2, Dst: 192.168.1.134  
 > User Datagram Protocol, Src Port: 53, Dst Port: 64042  
 > Domain Name System (response)

Figura 3.4: Port response a DNS

Quan accedim al servidor DNS, utilitzem el protocol **UDP** en lloc del protocol TCP perquè, en comparació amb TCP, UDP respon més ràpidament i, fins a cert punt, no necessitem garantir la seguretat de les dades. I UDP és més que suficient per fer servir els paquets utilitzats per accedir a DNS.

A la següent figura se'ns mostra que el procés del nostre accés al servidor *time-A* utilitza el protocol TCP:

No.	Time	Source	Destination	Protocol	Length	Info
89	9.166375	192.168.1.134	132.163.96.1	TCP	66	53356 → 13 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
95	9.334639	132.163.96.1	192.168.1.134	TCP	66	13 → 53356 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM
96	9.334748	192.168.1.134	132.163.96.1	TCP	54	53356 → 13 [ACK] Seq=1 Ack=1 Win=131328 Len=0
98	9.470854	132.163.96.1	192.168.1.134	DAYTIME	105	DAYTIME Response
99	9.470854	132.163.96.1	192.168.1.134	TCP	60	13 → 53356 [FIN, ACK] Seq=52 Ack=1 Win=65728 Len=0
100	9.470930	192.168.1.134	132.163.96.1	TCP	54	53356 → 13 [ACK] Seq=1 Ack=53 Win=131328 Len=0
101	9.473259	192.168.1.134	132.163.96.1	TCP	54	53356 → 13 [FIN, ACK] Seq=1 Ack=53 Win=131328 Len=0
102	9.608666	132.163.96.1	192.168.1.134	TCP	60	13 → 53356 [ACK] Seq=53 Ack=2 Win=65664 Len=0

Figura 3.5: Transmissió TCP

El següent diagrama ho pot mostrar encara amb més claredat:

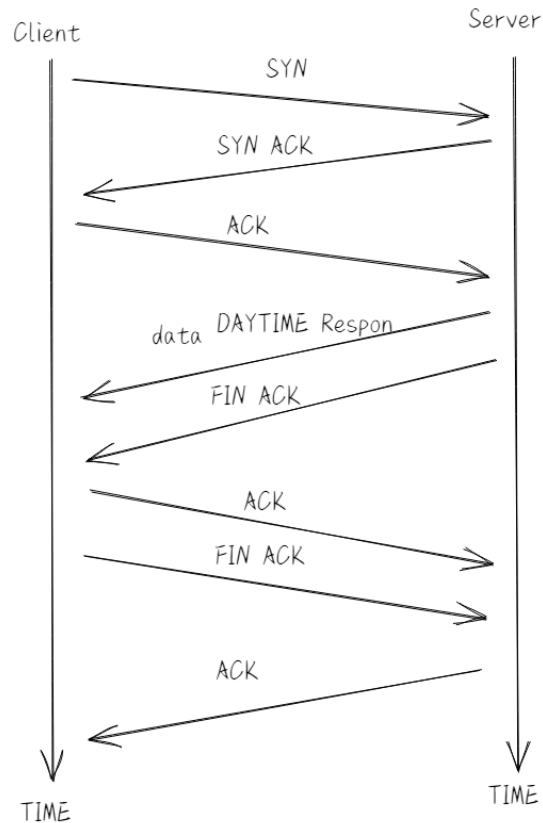


Figura 3.6: *Diagrama de transmissió TCP*

Flags a tenir en compte:

- **ACK:** Confirmar la recepció del missatge.
- **FIN:** Identificar a un paquet com l'últim de la connexió.
- **SYN:** Sincronitzar els nombres de seqüència inicials.

Primer, el client iniciarà una sol·licitud al servidor. Quan el servidor accepti la sol·licitud, donarà una resposta al client. Quan el client rebí la resposta, el servidor pot començar a enviar dades. Cada vegada que es rep una dada, el client pot donar una resposta al servidor. Diu que l'ha rebut, i quan s'envii l'últim missatge, es pot notificar al client perquè finalitzi la sessió, i quan el client rebí la sol·licitud del servidor per finalitzar la sessió, pot enviar una resposta, aleshores el servidor accepta la sessió.

El procés bàsic de TCP és així, a continuació hem de mirar els Flag del protocol **TCP**:

No.	Time	Source	Destination	Protocol	Length	Info
11	0.992291	192.168.1.134	132.163.96.1	TCP	66	54609 → 13 [SYN] Seq=0 W
15	1.132289	132.163.96.1	192.168.1.134	TCP	66	13 → 54609 [SYN, ACK] Seq
16	1.132349	192.168.1.134	132.163.96.1	TCP	54	54609 → 13 [ACK] Seq=1 A
20	1.272538	132.163.96.1	192.168.1.134	DAYTIME	105	DAYTIME Response
21	1.272538	132.163.96.1	192.168.1.134	TCP	60	13 → 54609 [FIN, ACK] Seq
22	1.272584	192.168.1.134	132.163.96.1	TCP	54	54609 → 13 [ACK] Seq=1 A

> Frame 11: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF\_{CAA3A2  
 > Ethernet II, Src: IntelCor\_9b:f5:eb (e4:5e:37:9b:f5:eb), Dst: zte\_c0:19:de (24:d3:f2:c0:19:de)  
 > Internet Protocol Version 4, Src: 192.168.1.134, Dst: 132.163.96.1  
 ✓ Transmission Control Protocol, Src Port: 54609, Dst Port: 13, Seq: 0, Len: 0  
 Source Port: 54609  
 Destination Port: 13  
 [Stream index: 5]  
 [Conversation completeness: Complete, WITH\_DATA (31)]  
 [TCP Segment Len: 0]  
 Sequence Number: 0 (relative sequence number)  
 Sequence Number (raw): 1596752081  
 [Next Sequence Number: 1 (relative sequence number)]  
 Acknowledgment Number: 0  
 Acknowledgment number (raw): 0  
 1000 .... = Header Length: 32 bytes (8)  
 ✓ Flags: 0x002 (SYN)  
 000. .... = Reserved: Not set  
 ...0 .... = Accurate ECN: Not set  
 .... 0... = Congestion Window Reduced: Not set  
 .... .0.. = ECN-Echo: Not set  
 .... ..0. = Urgent: Not set  
 .... ...0 = Acknowledgment: Not set  
 .... .... = Push: Not set  
 .... ..0.. = Reset: Not set  
 > .... .... .1. = Syn: Set  
 .... .... ..0 = Fin: Not set  
 [TCP Flags: .....S.]

Figura 3.7: *Flag SYN*

No.	Time	Source	Destination	Protocol	Length	Info
11	0.992291	192.168.1.134	132.163.96.1	TCP	66	54609 → 13 [SYN] Seq=0 W
15	1.132289	132.163.96.1	192.168.1.134	TCP	66	13 → 54609 [SYN, ACK] Seq
16	1.132349	192.168.1.134	132.163.96.1	TCP	54	54609 → 13 [ACK] Seq=1 A
20	1.272538	132.163.96.1	192.168.1.134	DAYTIME	105	DAYTIME Response
21	1.272538	132.163.96.1	192.168.1.134	TCP	60	13 → 54609 [FIN, ACK] Seq
22	1.272584	192.168.1.134	132.163.96.1	TCP	54	54609 → 13 [ACK] Seq=1 A

> Frame 15: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF\_{CAA3A2  
 > Ethernet II, Src: zte\_c0:19:de (24:d3:f2:c0:19:de), Dst: IntelCor\_9b:f5:eb (e4:5e:37:9b:f5:eb)  
 > Internet Protocol Version 4, Src: 132.163.96.1, Dst: 192.168.1.134  
 ✓ Transmission Control Protocol, Src Port: 13, Dst Port: 54609, Seq: 0, Ack: 1, Len: 0  
 Source Port: 13  
 Destination Port: 54609  
 [Stream index: 5]  
 [Conversation completeness: Complete, WITH\_DATA (31)]  
 [TCP Segment Len: 0]  
 Sequence Number: 0 (relative sequence number)  
 Sequence Number (raw): 786999761  
 [Next Sequence Number: 1 (relative sequence number)]  
 Acknowledgment Number: 1 (relative ack number)  
 Acknowledgment number (raw): 1596752082  
 1000 .... = Header Length: 32 bytes (8)  
 ✓ Flags: 0x012 (SYN, ACK)  
 000. .... = Reserved: Not set  
 ...0 .... = Accurate ECN: Not set  
 .... 0... = Congestion Window Reduced: Not set  
 .... .0.. = ECN-Echo: Not set  
 .... ..0. = Urgent: Not set  
 .... ...1 = Acknowledgment: Set  
 .... .... = Push: Not set  
 .... ..0.. = Reset: Not set  
 > .... .... .1. = Syn: Set  
 .... .... ..0 = Fin: Not set  
 [TCP Flags: .....A..S.]

Figura 3.8: *Flag SYN ACK*



No.	Time	Source	Destination	Protocol	Length	Info
11	0.992291	192.168.1.134	132.163.96.1	TCP	66	54609 → 13 [SYN] Seq=0 W
15	1.132289	132.163.96.1	192.168.1.134	TCP	66	13 → 54609 [SYN, ACK] Se
16	1.132349	192.168.1.134	132.163.96.1	TCP	54	54609 → 13 [ACK] Seq=1 A
20	1.272538	132.163.96.1	192.168.1.134	DAYTIME	105	DAYTIME Response
21	1.272538	132.163.96.1	192.168.1.134	TCP	60	13 → 54609 [FIN, ACK] Se
22	1.272584	192.168.1.134	132.163.96.1	TCP	54	54609 → 13 [ACK] Seq=1 A

> Frame 16: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF\_{CAA3A2  
 > Ethernet II, Src: IntelCor\_9b:f5:eb (e4:5e:37:9b:f5:eb), Dst: zte\_c0:19:de (24:d3:f2:c0:19:de)  
 > Internet Protocol Version 4, Src: 192.168.1.134, Dst: 132.163.96.1  
 ✓ Transmission Control Protocol, Src Port: 54609, Dst Port: 13, Seq: 1, Ack: 1, Len: 0  
   Source Port: 54609  
   Destination Port: 13  
   [Stream index: 5]  
   [Conversation completeness: Complete, WITH\_DATA (31)]  
   [TCP Segment Len: 0]  
   Sequence Number: 1 (relative sequence number)  
   Sequence Number (raw): 1596752082  
   [Next Sequence Number: 1 (relative sequence number)]  
   Acknowledgment Number: 1 (relative ack number)  
   Acknowledgment number (raw): 786999762  
   0101 .... = Header Length: 20 bytes (5)  
   ✓ Flags: 0x010 (ACK)  
     000. .... = Reserved: Not set  
     ...0 .... = Accurate ECN: Not set  
     ... 0... = Congestion Window Reduced: Not set  
     .... 0... = ECN-Echo: Not set  
     .... ..0. = Urgent: Not set  
     .... ...1 .... = Acknowledgment: Set  
     .... .... 0... = Push: Not set  
     .... .... ..0. = Reset: Not set  
     .... .... ..0. = Syn: Not set  
     .... .... ...0 = Fin: Not set  
   [TCP Flags: .....A....]

Figura 3.9: *Flag ACK*

No.	Time	Source	Destination	Protocol	Length	Info
11	0.992291	192.168.1.134	132.163.96.1	TCP	66	54609 → 13 [SYN] Seq=0 W
15	1.132289	132.163.96.1	192.168.1.134	TCP	66	13 → 54609 [SYN, ACK] Se
16	1.132349	192.168.1.134	132.163.96.1	TCP	54	54609 → 13 [ACK] Seq=1 A
20	1.272538	132.163.96.1	192.168.1.134	DAYTIME	105	DAYTIME Response
21	1.272538	132.163.96.1	192.168.1.134	TCP	60	13 → 54609 [FIN, ACK] Se
22	1.272584	192.168.1.134	132.163.96.1	TCP	54	54609 → 13 [ACK] Seq=1 A

> Frame 20: 105 bytes on wire (840 bits), 105 bytes captured (840 bits) on interface \Device\NPF\_{CAA3  
 > Ethernet II, Src: zte\_c0:19:de (24:d3:f2:c0:19:de), Dst: IntelCor\_9b:f5:eb (e4:5e:37:9b:f5:eb)  
 > Internet Protocol Version 4, Src: 132.163.96.1, Dst: 192.168.1.134  
 ✓ Transmission Control Protocol, Src Port: 13, Dst Port: 54609, Seq: 1, Ack: 1, Len: 51  
   Source Port: 13  
   Destination Port: 54609  
   [Stream index: 5]  
   [Conversation completeness: Complete, WITH\_DATA (31)]  
   [TCP Segment Len: 51]  
   Sequence Number: 1 (relative sequence number)  
   Sequence Number (raw): 786999762  
   [Next Sequence Number: 52 (relative sequence number)]  
   Acknowledgment Number: 1 (relative ack number)  
   Acknowledgment number (raw): 1596752082  
   0101 .... = Header Length: 20 bytes (5)  
   ✓ Flags: 0x018 (PSH, ACK)  
     000. .... = Reserved: Not set  
     ...0 .... = Accurate ECN: Not set  
     ... 0... = Congestion Window Reduced: Not set  
     .... 0... = ECN-Echo: Not set  
     .... ..0. = Urgent: Not set  
     .... ...1 .... = Acknowledgment: Set  
     .... .... 1... = Push: Set  
     .... .... ..0. = Reset: Not set  
     .... .... ..0. = Syn: Not set  
     .... .... ...0 = Fin: Not set  
   [TCP Flags: .....AP....]

Figura 3.10: *Flag data response*

No.	Time	Source	Destination	Protocol	Length	Info
11	0.992291	192.168.1.134	132.163.96.1	TCP	66	54609 → 13 [SYN] Seq=0 h
15	1.132289	132.163.96.1	192.168.1.134	TCP	66	13 → 54609 [SYN, ACK] Se
16	1.132349	192.168.1.134	132.163.96.1	TCP	54	54609 → 13 [ACK] Seq=1 A
20	1.272538	132.163.96.1	192.168.1.134	DAYTIME	105	DAYTIME Response
21	1.272538	132.163.96.1	192.168.1.134	TCP	60	13 → 54609 [FIN, ACK] Se
22	1.272584	192.168.1.134	132.163.96.1	TCP	54	54609 → 13 [ACK] Seq=1 A

> Frame 21: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF\_{CAA3A2  
 > Ethernet II, Src: zte\_c0:19:de (24:d3:f2:c0:19:de), Dst: IntelCor\_9b:f5:eb (e4:5e:37:9b:f5:eb)  
 > Internet Protocol Version 4, Src: 132.163.96.1, Dst: 192.168.1.134  
 > Transmission Control Protocol, Src Port: 13, Dst Port: 54609, Seq: 52, Ack: 1, Len: 0

Source Port: 13  
 Destination Port: 54609  
 [Stream index: 5]  
 [Conversation completeness: Complete, WITH\_DATA (31)]  
 [TCP Segment Len: 0]  
 Sequence Number: 52 (relative sequence number)  
 Sequence Number (raw): 786999813  
 [Next Sequence Number: 53 (relative sequence number)]  
 Acknowledgment Number: 1 (relative ack number)  
 Acknowledgment number (raw): 1596752082  
 0101 .... = Header Length: 20 bytes (5)  
 > Flags: 0x011 (FIN, ACK)  
 000. .... = Reserved: Not set  
 ...0 .... = Accurate ECN: Not set  
 .... 0... = Congestion Window Reduced: Not set  
 .... .0.. = ECN-Echo: Not set  
 .... ..0. = Urgent: Not set  
 .... ...1 .... = Acknowledgment: Set  
 .... ....0... = Push: Not set  
 .... .... .0.. = Reset: Not set  
 .... .... ..0. = Syn: Not set  
 > .... .... ...1 = Fin: Set  
 > [TCP Flags: .....A...F]

Figura 3.11: *Flag FYN*

A partir de les 5 imatges anteriors, podem observar 5 Flags de resposta diferents a TCP.

#### 4. EXERCICI 3

Al tercer exercici, utilitzarem el **ping** per iniciar una sol·licitud a [www.google.com](http://www.google.com), després utilitzarem *Wireshark* per capturar i analitzar la informació que hi ha a l'interior.

```
$ ping www.google.com

Haciendo ping a www.google.com [142.250.184.164] con 32 bytes de datos:
Respuesta desde 142.250.184.164: bytes=32 tiempo=13ms TTL=118
Respuesta desde 142.250.184.164: bytes=32 tiempo=14ms TTL=118
Respuesta desde 142.250.184.164: bytes=32 tiempo=13ms TTL=118
Respuesta desde 142.250.184.164: bytes=32 tiempo=20ms TTL=118

Estadísticas de ping para 142.250.184.164:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
    Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 13ms, Máximo = 20ms, Media = 15ms
```

Figura 4.1: *ping www.google.com*

A l'anterior figura, podem veure que hem fet 4 sol·licituds a [www.google.com](http://www.google.com) mitjançant la comanda **ping**. Les 4 peticions s'han contestat perfectament. També vam descobrir que la nostra adreça IP de [www.google.com](http://www.google.com) és **142.250.184.164**.

Al *Wireshark* de la figura següent, també hem trobat sol·licituds enviades des de la nostra adreça IP a l'adreça IP de *Google*, que eren 4 sol·licituds per nosaltres i 4 respostes enviades per *Google*.

Time	Source	Destination	Protocol	Length	Info
33.4.198359	192.168.1.129	192.258.184.164	ICMP	74	Echo (ping) request id=0x0001, seq=61/15616, ttl=128 (reply in 34)
34.4.131784	192.168.1.129	192.258.184.164	ICMP	74	Echo (ping) reply id=0x0001, seq=61/15616, ttl=128 (request in 33)
35.5.112969	192.168.1.129	192.258.184.164	ICMP	74	Echo (ping) request id=0x0001, seq=62/15617, ttl=128 (reply in 36)
36.5.127957	192.258.184.164	192.168.1.129	ICMP	74	Echo (ping) reply id=0x0001, seq=62/15617, ttl=128 (request in 35)
44.6.142618	192.168.1.129	192.258.184.164	ICMP	74	Echo (ping) request id=0x0001, seq=63/15622, ttl=128 (reply in 45)
45.6.137699	192.258.184.164	192.168.1.129	ICMP	74	Echo (ping) reply id=0x0001, seq=63/16128, ttl=128 (request in 44)
61.7.144188	192.168.1.129	192.258.184.164	ICMP	74	Echo (ping) request id=0x0001, seq=64/16384, ttl=128 (reply in 62)
62.7.164764	192.258.184.164	192.168.1.129	ICMP	74	Echo (ping) reply id=0x0001, seq=64/16384, ttl=128 (request in 61)
208.65.344666	192.168.1.129	192.258.184.164	QUIC	1292	Initial, DCID=4911ae9f9277e97e, PKN: 1, PING, PADDING, PING, PING, PING
209.65.345071	192.168.1.129	192.258.184.164	QUIC	117	0-RTT, DCID=4911ae9f9277e97e
210.65.382778	192.258.184.164	192.168.1.129	QUIC	292	Protected Payload (KP8)
211.65.382778	192.258.184.164	192.168.1.129	QUIC	225	Protected Payload (KP8)
212.65.383778	192.258.184.164	192.168.1.129	HTTP	833	Protected Payload (KP8)

Figura 4.2: *Informació capturada per Wireshark*

Entre ells, hem trobat una cosa nova: quan fem servir **ping**, fa servir el protocol **ICMP**.

[illegible]Figura 4.3: *Paquet de ICMP*

ICMP (*Internet Control Message Protocol*, Protocol de Missatges de Control d'Interxarxa) és un protocol que per al seu funcionament utilitza directament el protocol IP dins de l'arquitectura TCP/IP. La seva funció és informar de l'estat i situacions d'error en el funcionament de la capa de xarxa, sobretot en aspectes com l'encaminament, congestió, fragmentació, etc.

ICMP, al contrari que TCP i UDP, no s'utilitza directament per les aplicacions d'usuari. L'excepció és la comanda **ping**, que envia missatges de petició *Echo Request* (i rep missatges de resposta *Echo Reply*) per a determinar si un host està disponible i el temps que entren els paquets en anar i tornar a certa màquina.

```

Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0x4d1e [correct]
  [Checksum Status: Good]
  Identifier (BE): 1 (0x0001)
  Identifier (LE): 256 (0x0100)
  Sequence Number (BE): 61 (0x003d)
  Sequence Number (LE): 15616 (0x3d00)
  [Response frame: 34]
  Data (32 bytes)
    Data: 6162636465666768696a6b6c6d6e6f70717273747576776e16263646566676869
    [Length: 32]

```

Figura 4.4: *ICMP request*

```

Internet Control Message Protocol
  Type: 0 (Echo (ping) reply)
  Code: 0
  Checksum: 0x551e [correct]
  [Checksum Status: Good]
  Identifier (BE): 1 (0x0001)
  Identifier (LE): 256 (0x0100)
  Sequence Number (BE): 61 (0x003d)
  Sequence Number (LE): 15616 (0x3d00)
  [Request frame: 33]
  [Response time: 13,375 ms]
  Data (32 bytes)
    Data: 6162636465666768696a6b6c6d6e6f7071727374757677616263646566676869
    [Length: 32]

```

Figura 4.5: *ICMP response*

Les trames de *request* i *response* ICMP són similars. L'única diferència és que el tipus dels primers 7 bytes especifica la naturalesa del marc ICMP i els altres espais són els mateixos i opcionals.

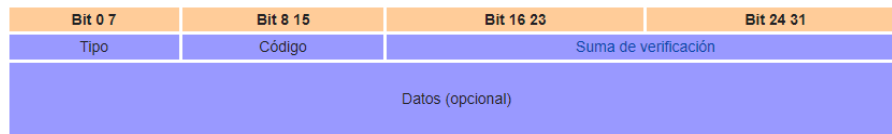


Figura 4.6: *Paquet ICMP*

A més, podem obrir la pàgina web de *Google* mitjançant **http://142.250.184.164**:

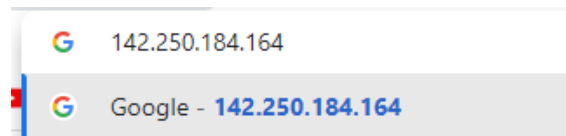


Figura 4.7: *Accedint a Google mitjançant l'IP*

Des de *Wireshark* a la següent figura, podem trobar que quan volem accedir al navegador, el port que fem servir és el **61016**, i el port d'accés que proporciona el servidor de *Google* és el port 80, que és el port predeterminat de la pàgina web del servidor.

(ip.addr == 192.168.1.129) and (ip.addr == 142.250.184.164)						
No.	Time	Source	Destination	Protocol	Length	Info
212	4.282335	192.168.1.129	142.250.184.164	TCP	55	61016 → 80 [ACK] Seq=1 Ack=1 Win=512 Len=1
213	4.282335	192.168.1.129	142.250.184.164	TCP	55	61017 → 80 [ACK] Seq=1 Ack=1 Win=512 Len=1
215	4.296846	142.250.184.164	192.168.1.129	TCP	66	80 → 61016 [ACK] Seq=1 Ack=2 Win=256 Len=0 SLE=1 SRE=2
216	4.296846	142.250.184.164	192.168.1.129	TCP	66	80 → 61017 [ACK] Seq=1 Ack=2 Win=256 Len=0 SLE=1 SRE=2

Figura 4.8: *Anàlisi de Wireshark a http://142.250.184.164*

Aquí, fem servir el protocol **TCP** per iniciar la sol·licitud al servidor de *Google* i també utilitzem el protocol **TCP** per controlar el flux.

## 5. EXERCICI 4

Al quart exercici, utilitzarem el *Packet Tracer* en més profunditat. Crearem dues xarxes, una xarxa estarà formada per una adreça estàtica, i l'altra xarxa estarà composta per DHCP i, a continuació, una de les xarxes estarà connectada a un servidor. La nostra elecció és que el servidor es crea a la xarxa estàtica i les dues últimes xarxes estan connectades entre si mitjançant el núvol.

El diagrama de topologia és el que es mostra a la figura següent, i l'adreça IP de cada ordinador i servidor està degudament marcada:

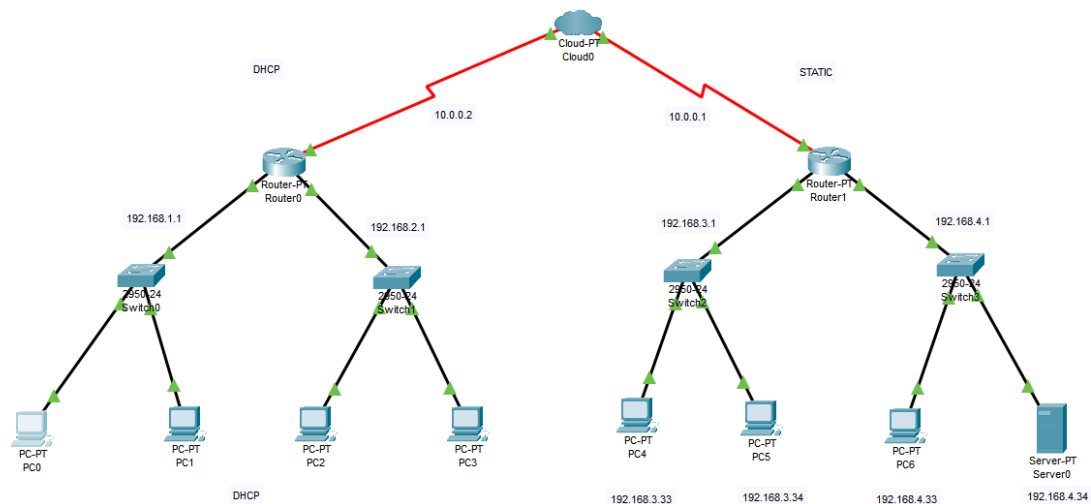


Figura 5.1: Diagrama de xarxa a Packet Tracer

Per tal de garantir i comprovar que les dues xarxes es connectin correctament a través del núvol, fem servir l'ordinador PC0 a la xarxa DHCP per enviar un paquet a l'ordinador PC4 a la xarxa estàtica.

A la següent figura, podem veure que les dues xarxes es poden connectar correctament:

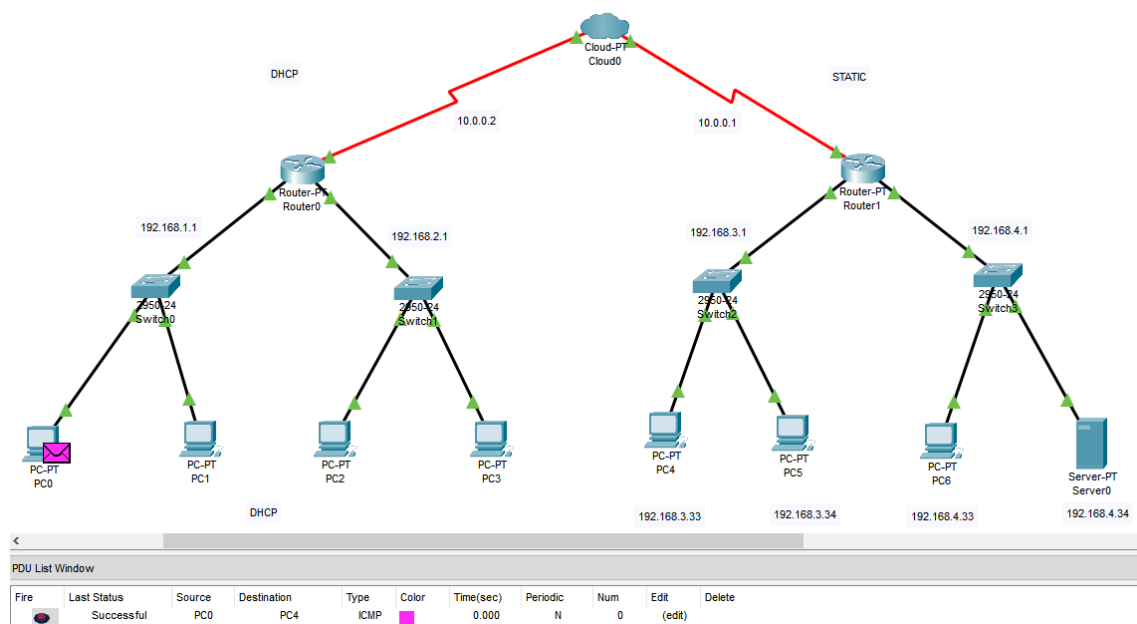


Figura 5.2: Connexió entre PC a DHCP i PC a STATIC

El mateix mètode s'utilitza per provar la comunicació entre PC0 i el servidor i permetre que el servidor envii una sol·licitud de resposta.

A partir dels resultats de les dues següents figures, podem veure que l'ordinador i el servidor es poden comunicar entre ells sota diferents xarxes.

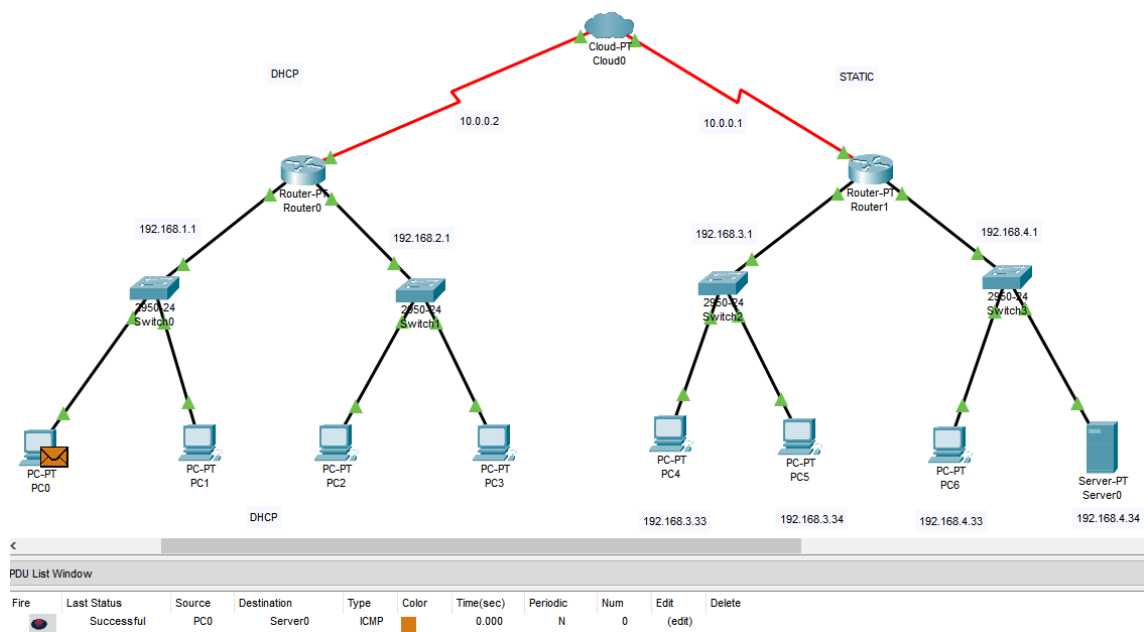


Figura 5.3: Connexió PC a server

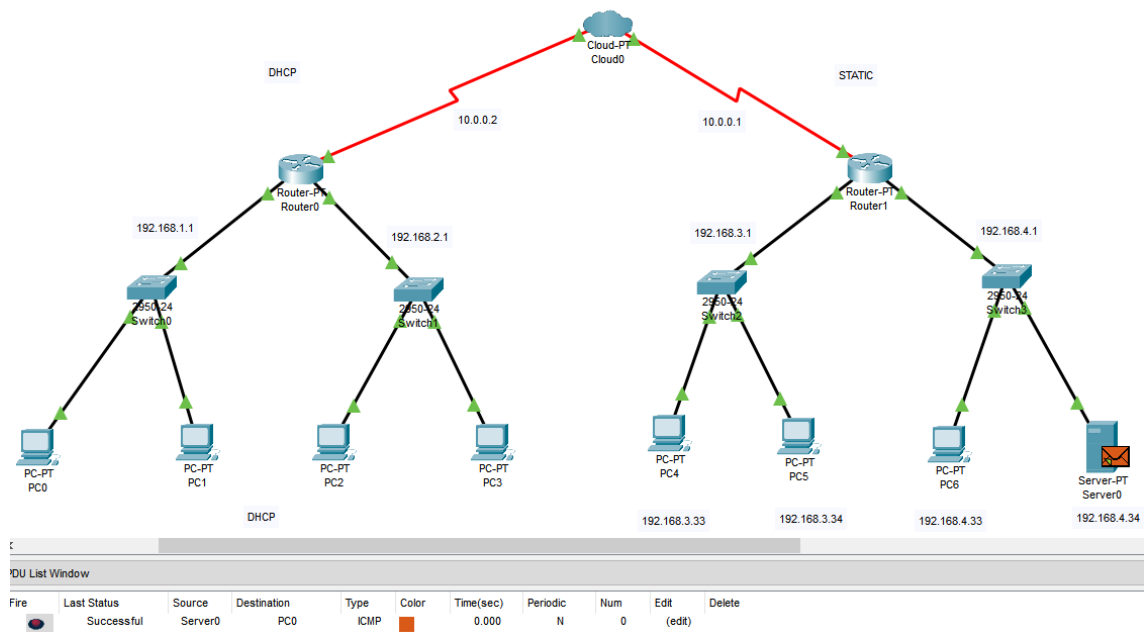


Figura 5.4: Connexió server a PC

Mitjançant les proves anteriors, hem garantit les funcions de comunicació entre ordinadors i servidors sota diferents xarxes. A continuació, provarem les funcions de les pàgines web del servidor a les quals accedeixen els ordinadors.

Com es pot veure a la imatge anterior, podem utilitzar perfectament el lloc web del servidor al qual accedeix l'ordinador de PC0.

A continuació, introduïrem alguns *frames* i protocols entre la comunicació entre PC i servidor.

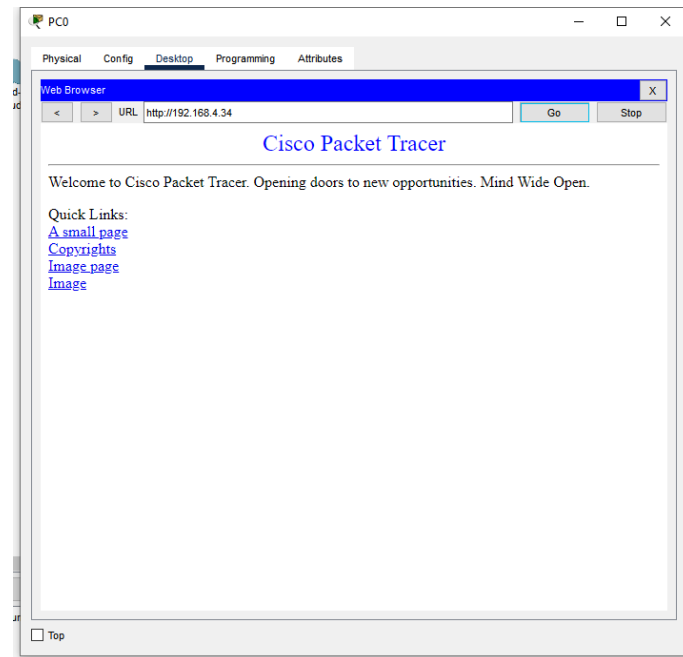


Figura 5.5: Connexió a pàgina web

En primer lloc, podem trobar que la comunicació entre l'ordinador i el servidor utilitza el protocol **TCP**.

Segons el nostre anàlisi del *flag* TCP al segon exercici, podem trobar que la primera sol·licitud iniciada pel PC al servidor és una sol·licitud **SYN**.

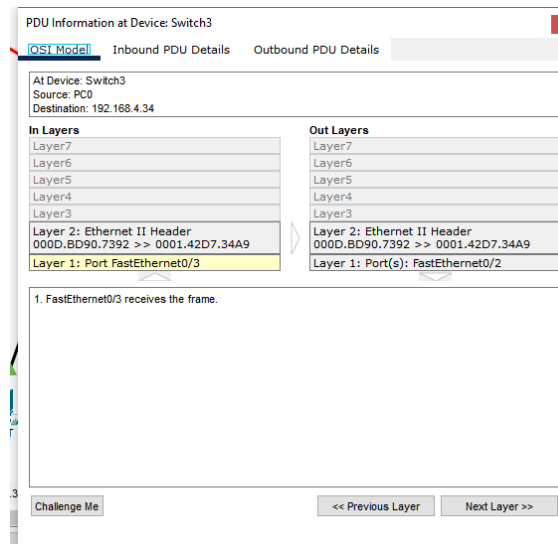


Figura 5.6: PDU PC request

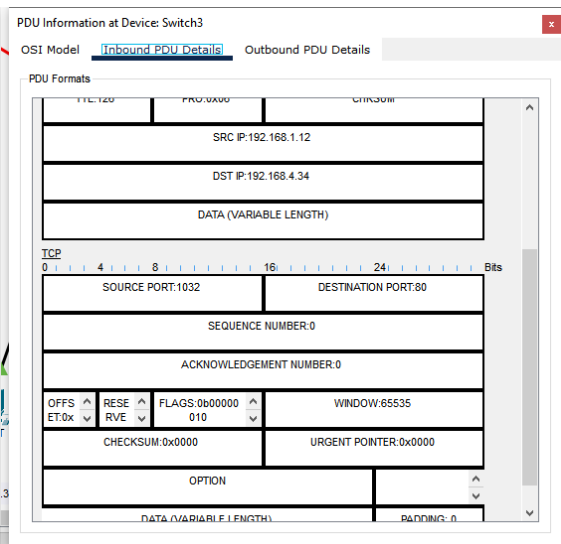


Figura 5.7: PDU PC request

De la mateixa manera, segons el nostre anàlisi del *flag* TCP al segon exercici, podem trobar que una possible resposta del servidor al PC és una resposta **ACK**:

També podem analitzar altres paquets de petició de la mateixa manera, però aquí analitzem un protocol que no s'ha vist al segon exercici.

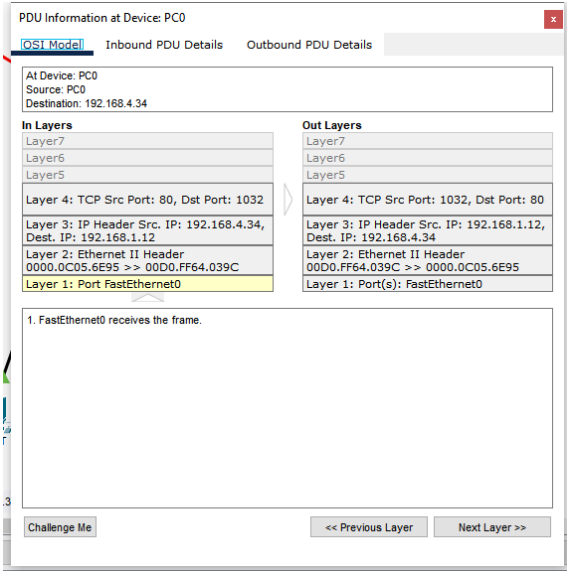


Figura 5.8: PDU PC response

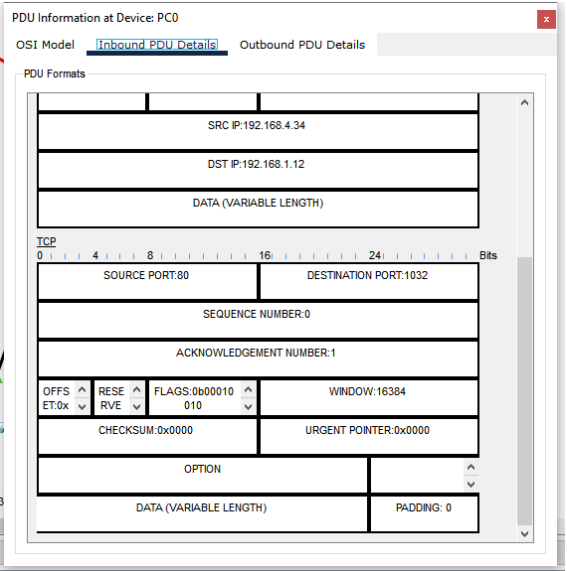


Figura 5.9: PDU PC response

Podem veure el protocol **HTTP**, el protocol de transferència d'hipertext o HTTP (*HyperText Transfer Protocol*) estableix el protocol per a l'intercanvi de documents d'hipertext i multimèdia al web. HTTP disposa d'una variant xifrada mitjançant SSL anomenada HTTPS.

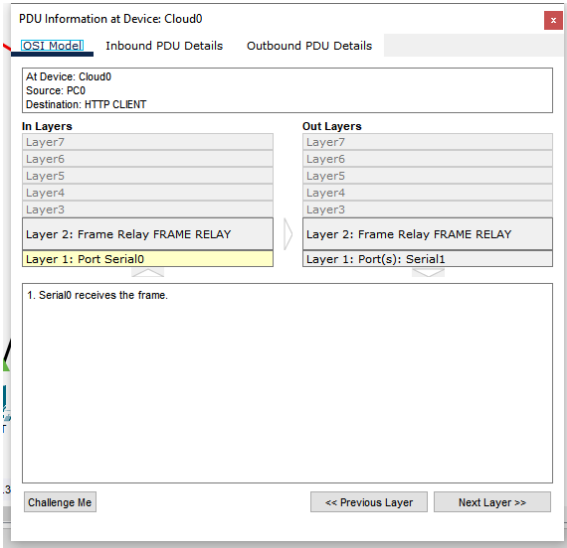


Figura 5.10: PDU HTTP request

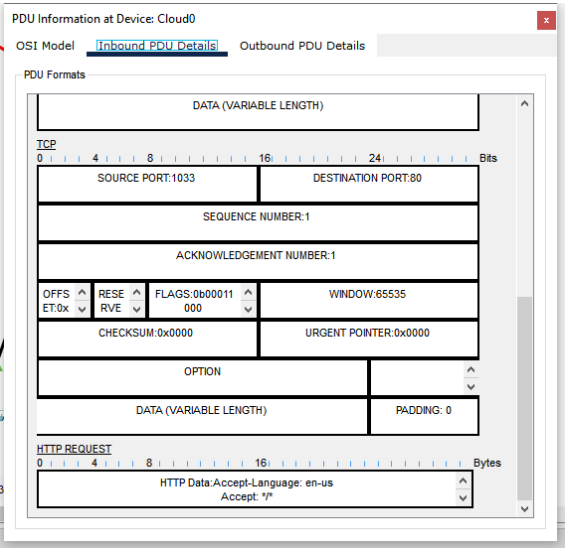


Figura 5.11: PDU HTTP request



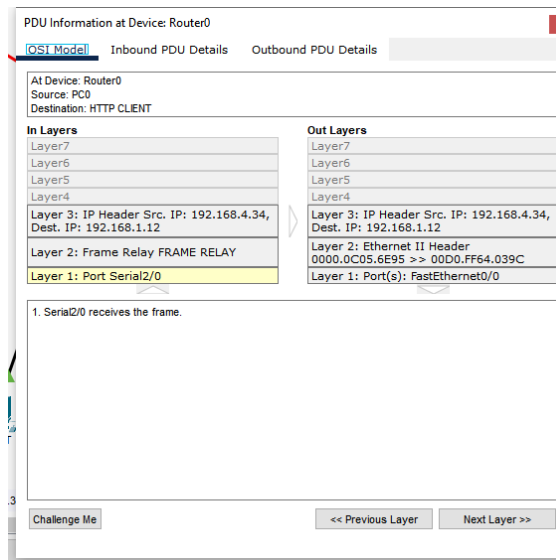


Figura 5.12: PDU HTTP response

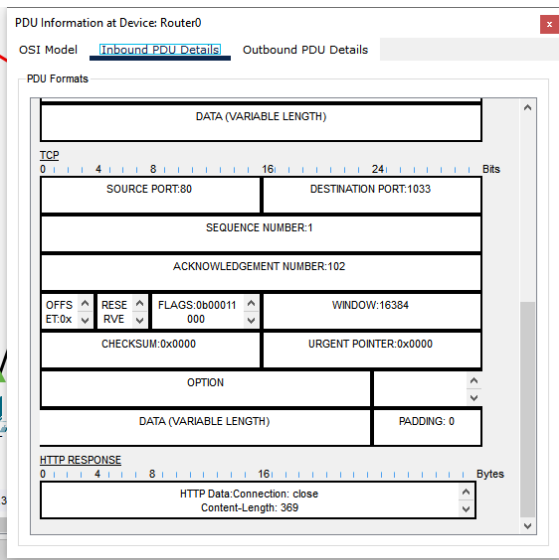


Figura 5.13: PDU HTTP response

Aquí podem veure que el protocol **TCP/IP + HTTP** s'utilitza quan l'ordinador i el servidor estableixen una sol·licitud web de connexió. Amb el que hem de prestar atenció aquí és que quan l'ordinador envia una sol·licitud web al servidor, fa servir **HTTP request**, i el servidor utilitza **HTTP response** quan vol que l'ordinador envii una resposta.

Per tal d'introduir millor la comunicació entre el PC i el servidor, fem servir un diagrama per presentar el seu procés de comunicació.

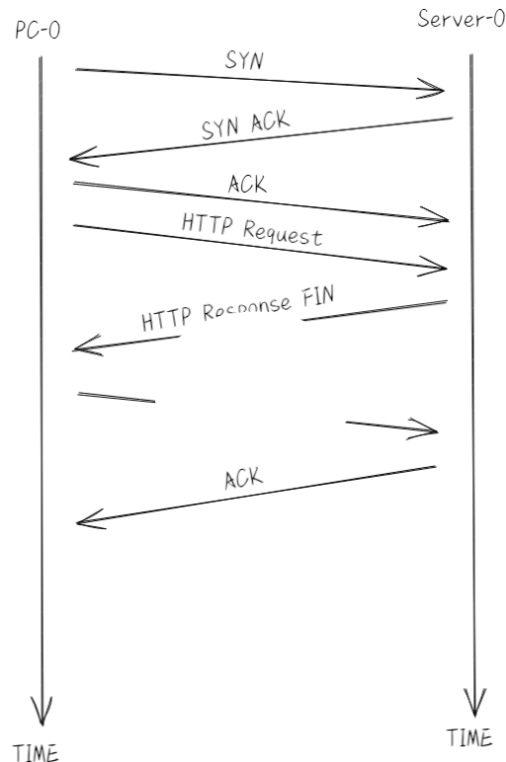


Figura 5.14: Diagrama de petició

Quan l'ordinador vulgui sol·licitar la web des del servidor, primer enviarà una sol·licitud de **SYN** al servidor, després el servidor respondrà a la sol·licitud (**SYN ACK**) i l'ordinador també respondrà **ACK**, assegurant així l'establiment de la comunicació entre l'ordinador i el servidor.

Quan l'ordinador confirmi l'establiment d'una sol·licitud amb el servidor, enviarà una sol·licitud **HTTP request**, i quan el servidor rebi la sol·licitud **HTTP request**, retornarà una sol·licitud **HTTP response**, un component principal del web.

Amb una sol·licitud d'alerta (**FIN**), vol dir que la sol·licitud **HTTP** s'ha completat. Quan l'ordinador rebi les peticions **HTTP response** i **FIN** enviades pel servidor, finalment demanarà al servidor que tanqui la comunicació, i quan el servidor rebi la sol·licitud de tancament de la comunicació, tancaran oficialment la sessió.

## 6. CONCLUSIONS

Aquesta pràctica ens ha servit molt per a acabar d'aprofundir en els conceptes estudiats i entendre en profunditat els diferents protocols esmentats a la pràctica. Hem après a utilitzar l'eina d'anàlisi de xarxa *Wiredshark* i hem analitzat la xarxa tenint una comprensió més profunda amb el model OSI. A més, també hem dissenyat i interconnectat diverses xarxes a través del núvol fent ús del *Packet Tracer*.