UNIVERSITAT DE BARCELONA

# Pràctica 4. El model OSI

Óscar Jiménez, NIUB: 20100286
Laurentiu Nedelcu, NIUB: 20081585

# ÍNDEX

## OBJECTIUS

L'objectiu principal d'aquesta pràctica és veure com s'encapsulen i desencapsulen les diferents Unitats de Protocol d'Usuari (DPU). A més, de com permeten transmetre informació entre dos equips de manera estàndard, independentment de les característiques dels equips. Per tal de dur a terme aquestes tasques farem servir el programa sniffer Wireshark.

## PREGUNTES I RESPOSTES

**Seleccioneu la interfície d'Ethernet. Apunteu l'adreça MAC que surt i executeu des de consola un ipconfig /all. Identifiqueu la IP associada a aquesta MAC. Descriviu curosament els detalls a l'informe.**

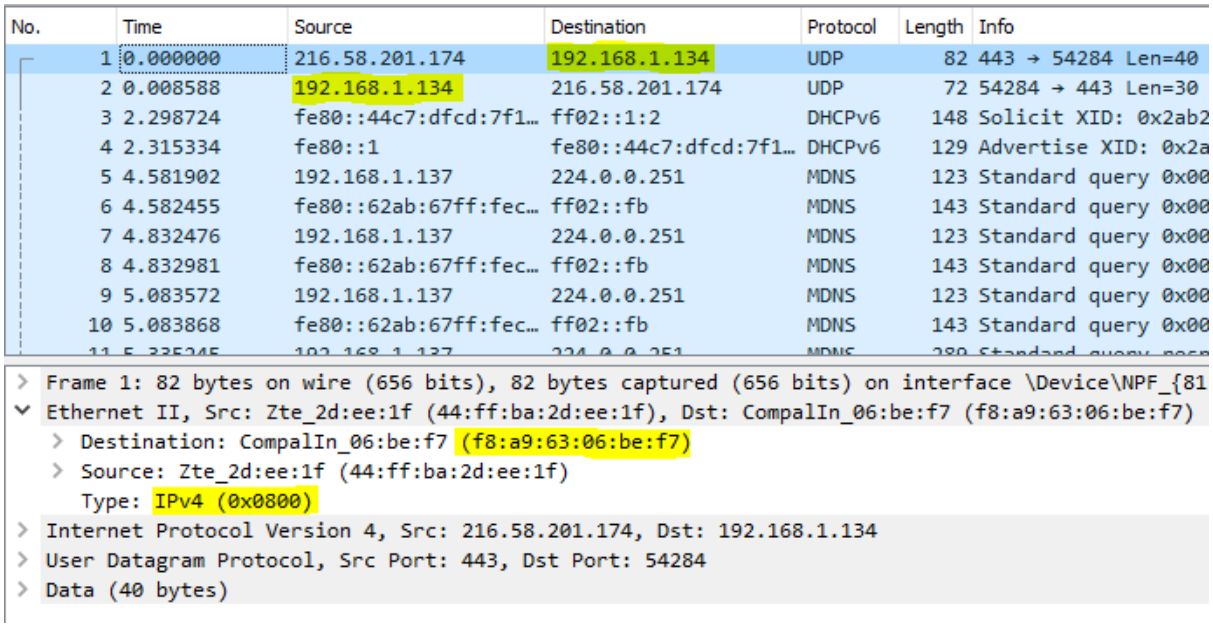
En primer lloc observem les dades del adaptador d'Ethernet que ens proporciona la terminal executant ipconfig /all. El que ens interessa en aquest cas és conèixer la nostra direcció física (adreça MAC) la qual és f8-a9-63-06-be-f7 e identificar les direccions IPs: 192.168.1.134 (IPv4) i fe80::44c7:dfcd:7f11:9a41%14 (IPv6).

```
Adaptador de Ethernet Ethernet:

   Sufijo DNS específico para la conexión. . : Home
   Descripción . . . . . . . . . . . . . . . : Broadcom NetLink (TM) Gigabit Ethernet
   Dirección física. . . . . . . . . . . . . : F8-A9-63-06-BE-F7
   DHCP habilitado . . . . . . . . . . . . . : sí
   Configuración automática habilitada . . . : sí
   Vínculo: dirección IPv6 local. . . : fe80::44c7:dfcd:7f11:9a41%14(Preferido)
   Dirección IPv4. . . . . . . . . . . . . . : 192.168.1.134(Preferido)
   Máscara de subred . . . . . . . . . . . . : 255.255.255.0
   Concesión obtenida. . . . . . . . . . . . : viernes, 27 de diciembre de 2019 12:23:16
   La concesión expira . . . . . . . . . . . : lunes, 30 de diciembre de 2019 12:28:45
   Puerta de enlace predeterminada . . . . . : 192.168.1.1
   Servidor DHCP . . . . . . . . . . . . . . : 192.168.1.1
   IAID DHCPv6 . . . . . . . . . . . . . . . : 116959587
   DUID de cliente DHCPv6. . . . . . . . . . : 00-01-00-01-1E-6F-8B-AD-F8-A9-63-06-BE-F7
   Servidores DNS. . . . . . . . . . . . . . : 212.231.6.7
                                       46.6.113.34
   NetBIOS sobre TCP/IP. . . . . . . . . . . : habilitado
```

*Imatge 1. Informació detallada de l'adaptador d'Ethernet de l'equip.*

A continuació, anem a trobar aquestes dades en les captures de les trames que ens proporciona el programa Wireshark. El primer que veiem és la nostra direcció IPv4 tant en la primera trama com en la segona. Aquesta es trobarà en "source" o "destination" depenent de si està enviant o si està rebent la informació. Si mirem amb més detall sobre la primera trama, veiem que en "destination" tenim la nostra adreça MAC (f8-a9-63-06-be-f7) i a més ens especifica quin tipus de protocol: IPv4. El mateix succeeix amb les trames 3 i 4 en les quals s'usa el IPv6 i aquesta també ens mostra la direcció física del nostre equip.

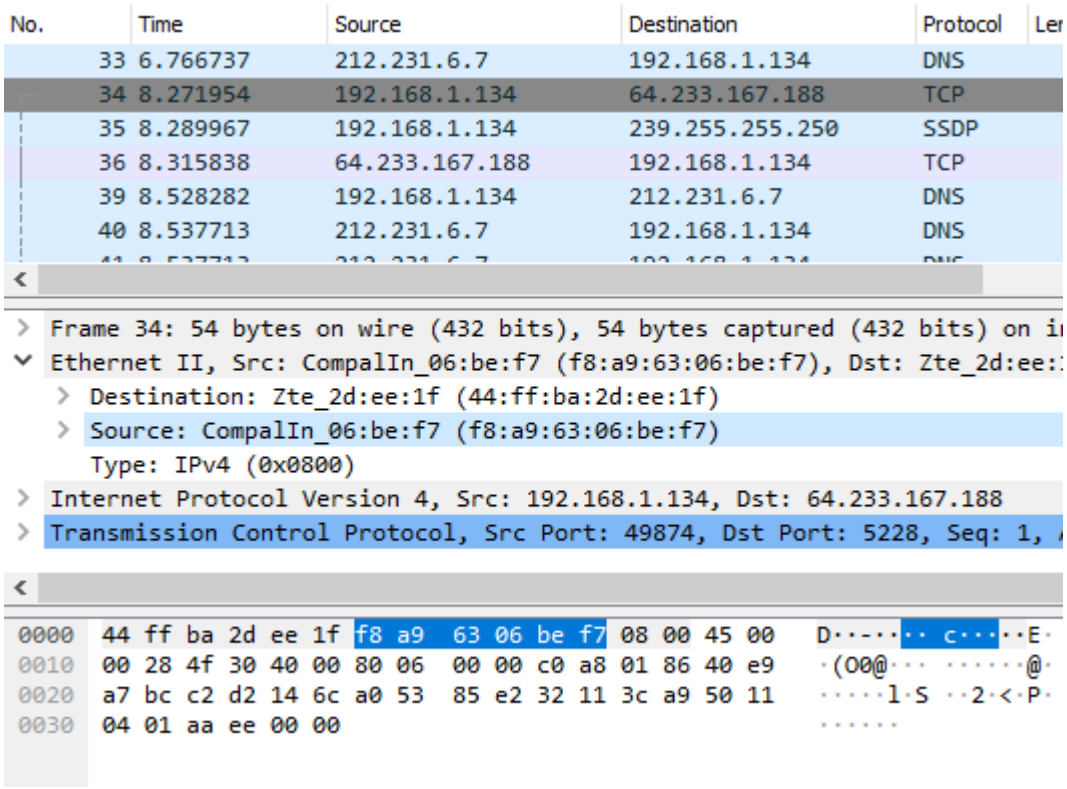



*Imatge 2. Informació mostrada de la captura d'Ethernet de l'equip. Ressaltat es veuen mostrades les dades explicades per al protocol IPv4.*

## Exercici 1

**Escolliu de la llista una de les captures on el protocol sigui TCP i seleccioneu-la. A la finestra horitzontal intermitja, es desglossa tot el paquet seleccionat. A la finestra inferior es desglossa el que teniu seleccionat a la finestra intermitja. Així, si cliqueu a Ethernet apareix la MAC vostra i la del destinatari. Com es descriu la vostra adreça MAC? A la adreça MAC hi han dues parts clarament diferenciades. A que corresponen? Compara el que apareix amb el que surt amb un ipconfig/all. Repassa els diferents camps que apareixen a la capçalera IP i amb l'ajut dels llibres i/o Internet identifica que fa cada un dels camps.**

La finestra inferior, conegut com a panel d'octets de paquets, ens mostra les dades del paquet actual (seleccionat de la "llista de paquets", finestra superior) en un estil hexdump. Cada línia conté el desplaçament del offset de les dades, en la part esquerra estan els 16 bytes en hexadecimal i en la part esquerra veiem els 16 bytes en ASCII. Els bytes ASCII que no es poden imprimir es substitueixen per un període (".”). La nostra MAC es pot veure en la imatge inferior.




*Imatge 3. Les tres finestres descrites en l'enunciat, en la finestra inferior es veu la nostra adreça MAC.*

Veiem que la nostra adreça MAC en hexadecimal, és la que hem vist anteriorment, i en ASCII tenim (".. c…").

Aquesta està determinada i configurada en dues parts. La primera meitat (el primers 24 bits) indica el fabricant del dispositiu del hardware. Aquests identificadors estan regulats per l'Autoritat de Números Assignats d'Internet (IANA). Per l'altra banda, la segona meitat (els últims 24 bits) indica el número de sèrie del dispositiu individual regulat per l'Institut d'Enginyers Elèctrics i Electrònics (IEEE).

En quant als diferents camps que ens apareixen en la capçalera IPv4 són els que es veuen en la imatge 4. El protocol d'internet (IP) està definit en RFC 791.

El primer que veiem és la versió (4 bits) de l'IP el qual indica el format de la capçalera, en aquesta captura la versió és 4.

El següent camp és la longitud de capçalera d'internet (4 bits) que és la longitud de l'encapçalament d'internet en blocs de 32 bits, i apunta al començament de les dades. Per tant, la mida més gran d'aquest camp són de **60 bytes**, ja que el valor màxim del IHL és de 15 (en binari 1111), i si fem 15 x 32 = 480 bits = 60 bytes. Tindre en compte que la mida mínima és de **20 bytes**, el valor que tenim en aquest cas (5 x 32 = 160 bits = 20 bytes). Això, es déu a què tots els camps obligatoris de la capçalera necessiten 20 bytes d'espai.

```
Internet Protocol Version 4, Src: 192.168.1.134, Dst: 64.233.167.188
   0100 .... = Version: 4
   .... 0101 = Header Length: 20 bytes (5)
 ⌄ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      0000 00.. = Differentiated Services Codepoint: Default (0)
      .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
   Total Length: 40
   Identification: 0x4f30 (20272)
 ⌄ Flags: 0x4000, Don't fragment
      0... .... .... .... = Reserved bit: Not set
      .1.. .... .... .... = Don't fragment: Set
      ..0. .... .... .... = More fragments: Not set
   ...0 0000 0000 0000 = Fragment offset: 0
   Time to live: 128
   Protocol: TCP (6)
   Header checksum: 0x0000 [validation disabled]
   [Header checksum status: Unverified]
   Source: 192.168.1.134
   Destination: 64.233.167.188
```

*Imatge 4. Informació detallada dels diferents camps que hi apareixen en la capçalera del protocol d'internet.*

En el tercer camp tenim el tipus de servei el qual proporciona una indicació del resum dels paràmetres de la qualitat del servei desitjat. Aquests paràmetres s'utilitzen per orientar la selecció dels paràmetres de servei, que d'alguna manera considera el tràfic de precedència més alta com a més important que d'altres trànsits. Els valors dels bits d'aquest camp tenen els següents significats:

```
Bits 0-2:  Precedence.
Bit    3:  0 = Normal Delay,      1 = Low Delay.
Bits   4:  0 = Normal Throughput, 1 = High Throughput.
Bits   5:  0 = Normal Relibility, 1 = High Relibility.
Bit  6-7:  Reserved for Future Use.

   0     1     2     3     4     5     6     7
+-----+-----+-----+-----+-----+-----+-----+-----+
|                 |     |     |     |     |     |
|   PRECEDENCE    |  D  |  T  |  R  |  0  |  0  |
|                 |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+

  Precedence

    111 - Network Control
    110 - Internetwork Control
    101 - CRITIC/ECP
    100 - Flash Override
    011 - Flash
    010 - Immediate
    001 - Priority
    000 - Routine
```

*Imatge 5. Informació detallada dels diferents valors que pot tindre els bits del camp del tipus de servei.*

Nosaltres, en l'exemple donat tenim els 8 bits posat a 0, per tant, tenim seleccionat l'opció de rutina i el retard, rendiment i fiabilitat en mode normal. Als bits reservats no li donem ús.

La longitud total (16 bits) és la longitud del datagrama, mesurada en octets, incloses les capçaleres i dades d'internet. Aquest camp permet la longitud d'un datagrama de fins a 65535 octets. Tot i això, aquests datagrames llargs són poc pràctics per a la majoria dels hostes i xarxes, ja que els aquests estan preparats per acceptar datagrames de fins a 576 octets (tant si arriben sencers o en fragments). En el cas de que un hoste vulgui transmetre datagrames més grans, han d'estar segurs que el destinatari pugui rebre aquests datagrames tan llargs.

El valor de l'identificador (16 bits) té la finalitat d'ajudar a muntar els fragments dels datagrames enviats.

El significat dels valors de les banderes (3 bits) es poden veure en la imatge inferior. Mitjançant aquests valors podem saber que el nostre datagrama no està fragmentat. I per tant, no s'haurà d'esperar a que arribin més fragments per tal de completar el datagrama.

```
Bit 0: reserved, must be zero
Bit 1: (DF) 0 = May Fragment,  1 = Don't Fragment.
Bit 2: (MF) 0 = Last Fragment, 1 = More Fragments.

    0   1   2
  +---+---+---+
  |   | D | M |
  | 0 | F | F |
  +---+---+---+
```

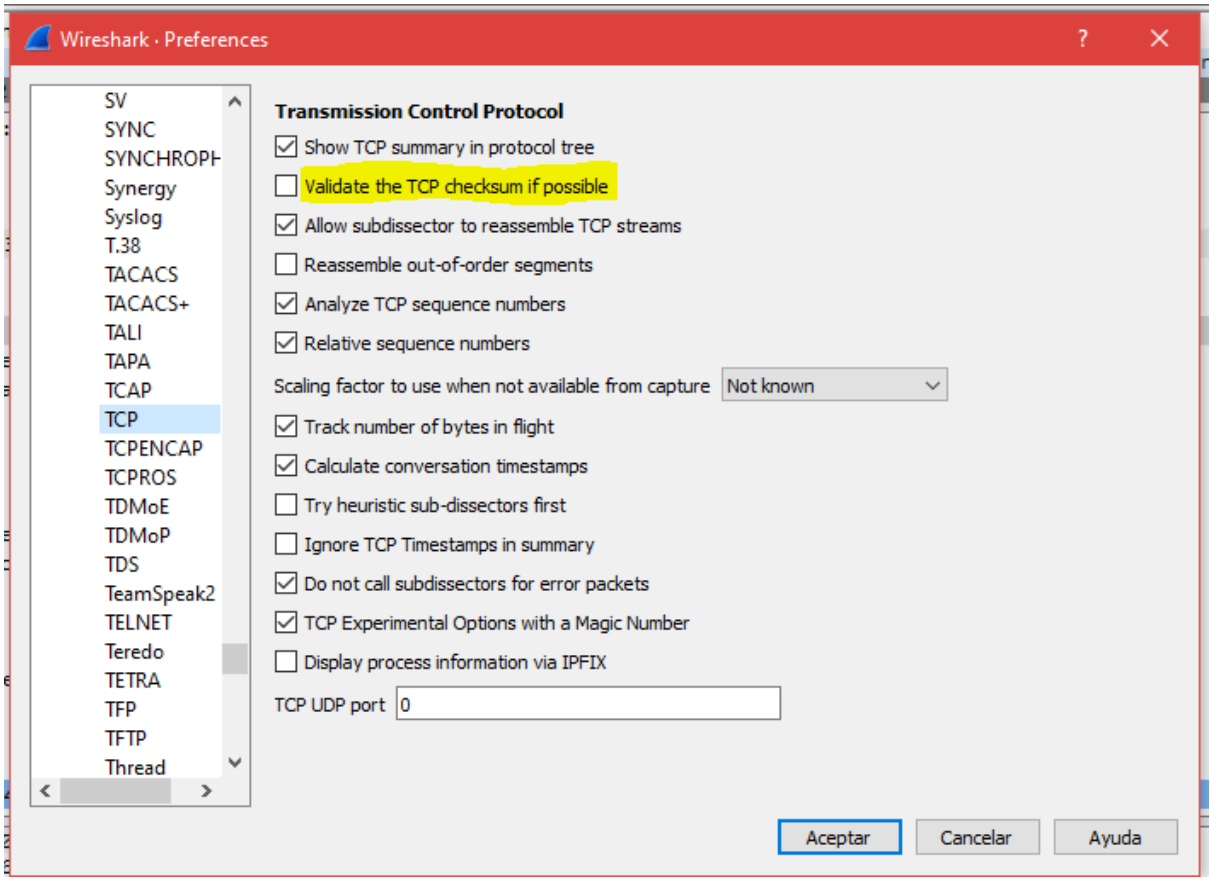
*Imatge 6. Només es poden usar 2 dels possibles 3 bits, ja que el primer està reservat.*

El camp del "Fragment Offset" (13 bits) ens indica on pertany aquest fragment en el datagrama. Aquest camp és mesura en unitats de 8 octets (64 bits). En el nostre cas el fragment el qual s'ha enviat té el valor 0. Si pensem, aquest valor té sentit ja que és el primer fragment, i per tant, anirà al principi del datagrama. A més, recordem que aquest datagrama no té més fragments.

El camp del temps de vida "TTL" (8 bits) indica el temps màxim en què és permet el datagrama romandre al sistema d'internet. Si aquest camp conté el valor 0, aleshores s'ha de destruir aquest datagrama. El temps està en unitats de segon. Per al nostre datagrama tenim 128 segons.

El camp del protocol (8 bits) indica el protocol que es fa servir a les porcions de les dades del datagrama d'internet. Els valors de diversos protocols estan especificats en el RFC 790, "Assigned Numbers". Nosaltres tenim el protocol TCP, tal i com es demana en l'enunciat.

El camp de la capçalera del checksum (16 bits) conté la comprovació de la suma en només la capçalera ja que alguns camps canvien, com ara el TTL. Aquests es re-executen i es verifiquen en cada punt que es processi la capçalera d'internet. Ens hem donat compte que en les nostres captures no s'ha calculat el checksum ja que teníem l'opció desactivada en el Wireshark. És per això, que en la imatge 4 apareix que el checksum és 0 i al costat "validation disabled", a l'igual que al header checksum status.




*Imatge 7. L'opció de la validació del checksum està desactivada. Per conseqüència no podem saber si el paquet està corromput.*

I per últim, tenim els camps de les adreces origen i destí, tenen 32 bits cadascuna. Veiem que el qui transmet el datagrama és el nostre equip ja que conté la nostra direcció IPv4, i el qui rep conté una altra direcció IPv4. Per veure més informació sobre les adreces veure el següent enllaç: https://tools.ietf.org/html/rfc791#section-3.2.

## Exercici 2

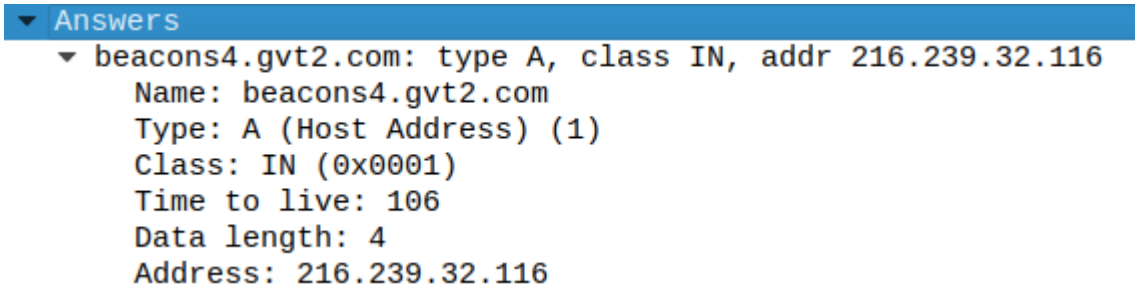
**Un cop determinada la IP, a quin port ens estem connectant? Quin protocol de transport fem servir?**

Tal i com li estem indicant per consola ens estem connectant al port 13. El port destinatari (el nostre) és el port 42558. Estem usant el protocol Daytime el qual és un protocol de comunicacions entre computadors que usa el port 13 (TCP i UDP). Aquest protocol està definit en el RFC 867.

**Identifica l'intercanvi de comunicacions que es produeix a nivell de DNS. Quin protocol de transport fa servir DNS? Perquè? Quina és la IP del servidor de DNS? Com s'especifica la resposta? Què respon?**

Normalment el protocol DNS fa servir el protocol UDP com a transport ja que és molt més ràpid. Tot i així, en ocasions usa el protocol TCP, això es dóna quan es necessita transportar respostes majors a 512 bytes de longitud i quan s'intercanvia informació entre servidors, per raons de fiabilitat. El servidor de DNS que se'ns mostra és 127.0.0.1.

Aquest protocol ens retorna la direcció IP del domini al qual volem accedir, la resposta es pot veure en "Answers" el qual té els següents camps:




*Imatge 8. Encara que "Answers" té més camps, ens centrarem en els que pensem que són de més rellevància.*

El camp "Name" que ens indica el nom del domini al qual volem accedir. El "Name" s'interpreta de dreta a esquerra i tracta el període (".") com a delimitador per a cada part del nom. Per exemple, prenem com a referència el domini midns.com. Veiem que aquest domini existeix al domini de nivell superior ".com". Després d'això, tenim el subdomini "midns" i no hi tenim més delimitadors, per tant, no hi ha més subdominis. Aquesta part es veurà millor explicada en la següent pregunta.

El camp de recursos "Type" és on es defineix el format del registre. Hi ha diferents tipus de registres de recursos, els més comuns són: A (el que tenim en aquest cas i que dóna una adreça IPv4), AAAA (que dóna una adreça IPv6), MX (que estableix la ubicació d'un servidor de correu), TXT (que pot incloure qualsevol text arbitrari). Aquest camp és el que defineix quina mena de "RData" cal esperar per al registre.

EL camp "Class" indica quina classe de consulta estem fent. Aquest camp no s'utilitza sempre ja que pràcticament sempre estarà en "IN" (indica que aquest registre és de la classe "Internet") en una resposta DNS. Altres classes podrien ser CH (per a Chaosnet), HS (Hesiod) i altres opcions QCLASS per a l'ús només de consultes.

El següent camp TTL ja el coneixem i ja l'hem explicat en l'exercici 1.

El "Data Length" és la longitud de la resposta el qual té una longitud de 32 bits que equivaldria als 4 bytes que hi apareixen en la imatge.

I per últim, tenim el camp "RData" o simplement "Address" el qual és la resposta que ens interessa, en aquest cas concret que estem buscant un registre A, ens està donant una direcció IPv4 que indica on està el domini. Depenent del registre (camp "Type") que busquem tindrem un RData diferent.

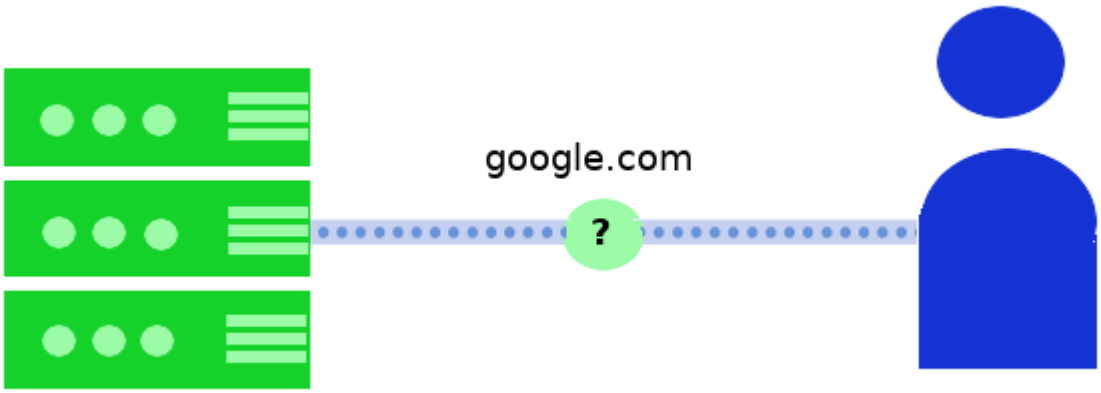
**Feu el diagrama temporal descrivint detalladament l'intercanvi d'informació entre el vostre ordinador i el servidor de DNS.**

El sistema de noms de domini (DNS) ens tradueix els noms de domini (com ara google.com) en identificadors binaris (adreces IP) associats als equips connectats a la xarxa, amb el propòsit de poder localitzar i direccionar aquests equips mundialment.

Aquest utilitza una base de dades distribuïda i jeràrquica que emmagatzema informació associada a noms de domini en xarxes com ara Internet.

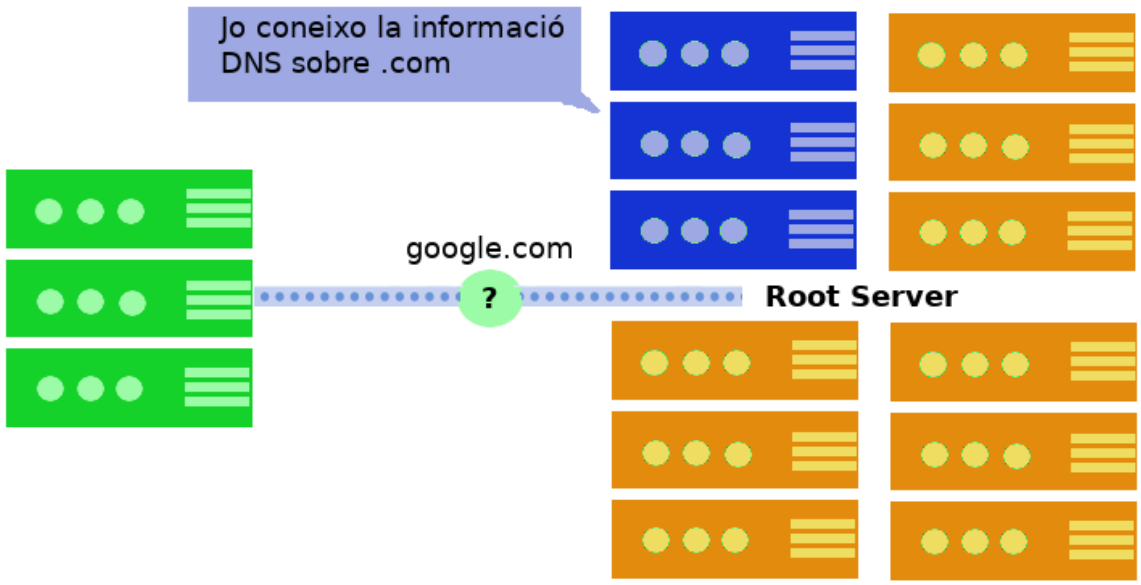
Per tal d'explicar com funciona l'intercanvi d'informació entre el nostre ordinador i el servidor de DNS seguirem els següents passos:

En primer lloc, quan busquem una pàgina web com ara (google.com) en el nostre buscador web, el nostre navegador envia una consulta per Internet per tal de trobar-la. El primer servidor amb el qual interactua la consulta és el **solucionador recursiu**, que pot ser gestionat pel Proveïdor de Serveis d'Internet (ISP), per l'operador de telefonia sense fils o un proveïdor de tercers.




*Imatge 9. Demanem al solucionador recursiu que ens trobi el nostre domini.*

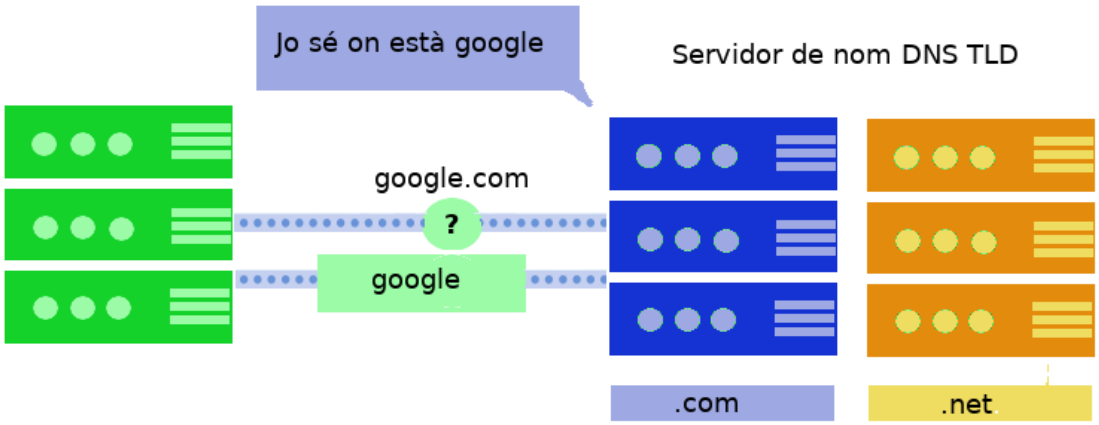
El primer tipus de servidor DNS al qual crida el solucionador recursiu és el Root Server. Els Root Servers s'executen arreu del món i cadascun coneix informació DNS sobre els dominis de nivell més alt (TLD), com ara ".com". Per tant, el solucionador recursiu demana a un servidor informació DNS sobre ".com".




*Imatge 10. El solucionador recursiu consulta al Root Server i aquest li proporciona la informació sobre ".com".*

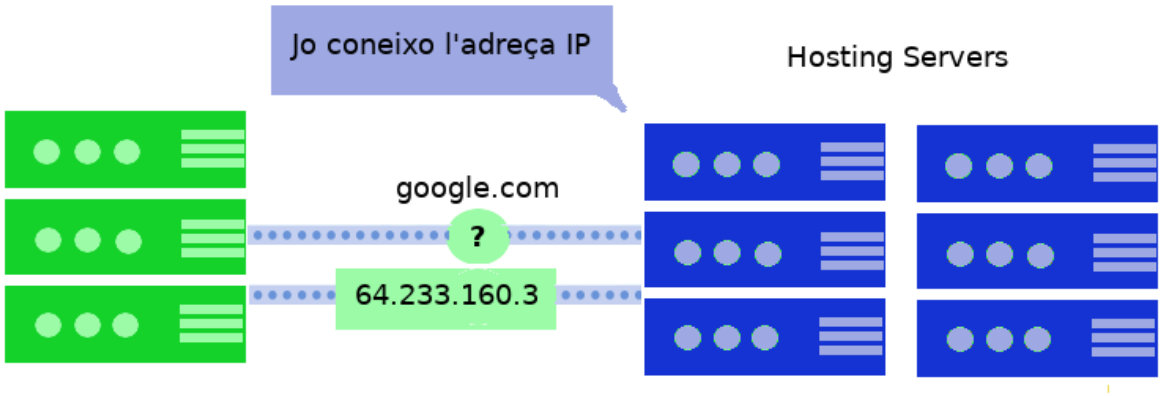
Hi ha milers de servidors donant suport al Root, localitzats de tal manera on més demanda d'Internet hi hagi. El DNS s'assegura que la nostra consulta serà enviada a un servidor que no estigui molt lluny de la nostra localització.

Cada servidor de nom DNS TLD emmagatzema la informació de les adreces de dominis de segon nivell dins del domini de primer nivell. Quan la consulta arriba al servidor TLD, el servidor respon amb l'adreça del servidor del nom del domini.



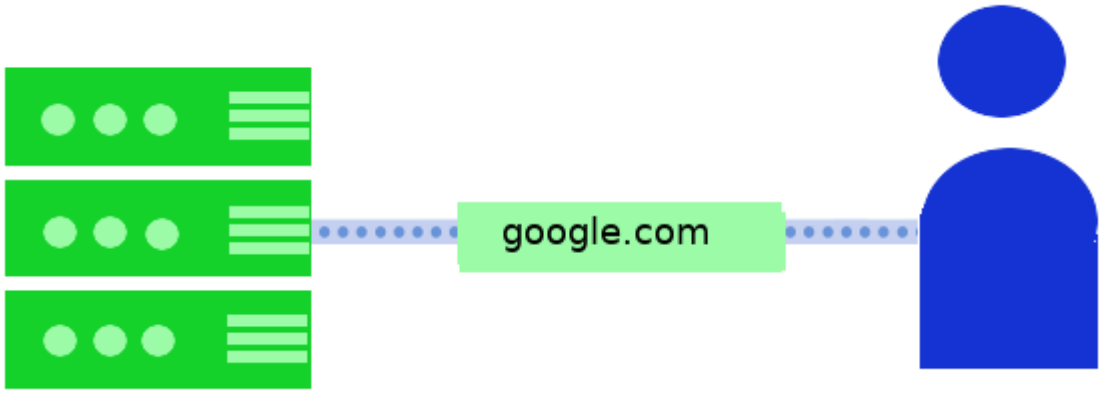

*Imatge 11. El servidor de nom TLD localitza google.com i li notifica al nostre solucionador.*

A continuació, el solucionador recursiu envia la consulta als Hosting Servers. Aquest servidor DNS coneix l'adreça IP del domini (google.com) i aquesta resposta és donada al solucionador.




*Imatge 12. El solucionador consulta la direcció IP del nom del domini i el Hosting Server li hi proporciona.*

Ara que el solucionador recursiu sap l'adreça IP del domini al qual hem fet la consulta, aquest li comunica al navegador la direcció. Finalment, el navegador pot transmetre una sol·licitud a la pàgina web per mostrar el contingut de la pàgina web amb l'adreça IP que se li ha atorgat amb el DNS.




*Imatge 13. El solucionador recursiu li mostra la pàgina web a l'usuari.*

**Un cop coneguda la IP destí, proporcionada pel servidor de DNS, identifica l'intercanvi de control que es produeix a nivell de TCP per la transmissió de la informació. Expliqueu que fa cada paquet i feu un diagrama temporal on es representa aquest intercanvi. Pren molta rellevància la utilització dels flags a TCP. Indiqueu que fan i com es fan servir per gestionar la comunicació.**

Les datagrames capturades amb les direccions IPs que ens proporciona el servidor de DNS utilitzen protocol UDP. Per tant, usarem una altra adreça IP el qual si utilitzi el protocol TCP, per tal d'explicar la transmissió de la informació. Més concretament usarem l'adreça 132.163.96.1 que és la direcció IP de telnet time-A.timefreq.bldrdoc.gov 13, el que se'ns proporciona en aquets exercici. Pensem que el motiu de perquè el servidor DNS usa el protocol UDP enlloc del TCP és, com ja hem dit abans, perquè el transport UDP treballa molt més ràpid i la mida de les dades són petites.

Abans de començar, s'ha de dir que mostrarem només les dades més rellevants en el diagrama i no tot el que ens mostra el wireshark ja que sinó quedarà molt carregat. A l'enunciat se'ns demana usar especialment els flags del TCP. Com ja hem dit usarem les dades amb més rellevància, per tant en el diagrama es veuran només els flags: SYN, ACK, PSH y FIN.

El flag SYN s'utilitza en el primer pas de la fase d'establiment de la connexió. Només el primer paquet de l'emissor i el receptor haurien de tenir establert aquest indicador. Aquesta etiqueta s'usa per sincronitzar el número de la seqüència, és a dir, per indicar a l'altre extrem quin número de seqüència han d'exceptuar.

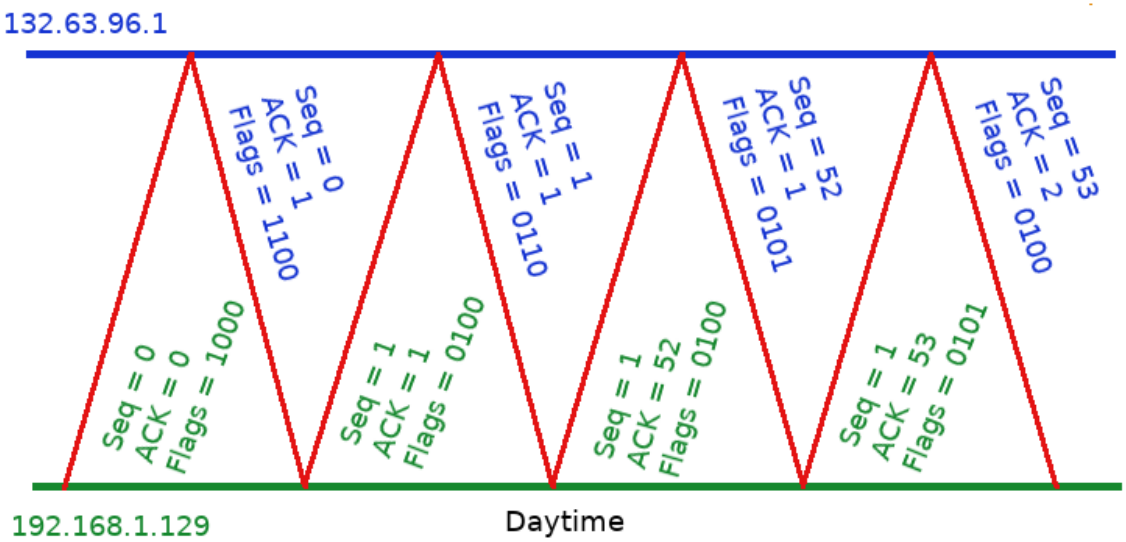
El flag ACK s'utilitza per tal de reconèixer els paquets que el host ha rebut amb èxit. El marcatge s'estableix si el camp del número d'acceptació conté un número de reconeixement vàlid.

El flag PSH indica que les dades de este segment i les dades que hagin sigut emmagatzemades anteriorment en el buffer del receptor s'han de transferir a l'aplicació receptora el més aviat possible. De vegades arriben diversos segments que transporten dades i no tenen activat el bit PSH, aleshores el receptor emmagatzemarà aquestes dades però no les entregarà a l'aplicació receptora fins que rebi un segment amb el PSH activat.

El flag FIN s'utilitza per sol·licitar la finalització de la connexió, és a dir, quan ja no hi ha més dades del remitent, sol·licita la finalització de la connexió. Aquest és l'últim paquet enviat i amb això s'allibera els recursos reservats i es finalitza la connexió.

Altres etiquetes importants però que no hem usat ja que el valor d'aquests no varia al llarg del temps són: el flag RST el qual s'usa per finalitzar la connexió si l'emissor RST veu que alguna cosa no funciona amb la connexió TCP o que aquesta connexió no hauria d'existir, i el flag URG que indica que el segment amb aquesta etiqueta s'envia de forma immediata a la capa d'aplicació, fins i tot si hi ha més dades a proporcionar a aquesta cap. S'utilitza per notificar al receptor el processament dels paquets urgents abans de processar tots els altres. El receptor serà notificat quan s'hagin rebut totes les dades urgents conegudes.

A més de les etiquetes esmentades es veuran el **número de la seqüència** i del **acknowledge**. L'**ordre de les etiquetes** en el diagrama són en les que hem comentat **(SYN, ACK, PSH, FIN)**.




*Imatge 14. Diagrama temporal de la transmissió de paquets entre dos hosts.*

El que veiem en el diagrama és que el nostre host (192.168.1.129) intenta establir una connexió TCP amb la IP esmentada anteriorment. Per a que això sigui possible, el receptor espera la connexió executant les primitives LISTEN (escolta el canal per tal de detectar una connexió) i ACCEPT (acceptar una connexió). El nostre host executa la primitiva CONNECT (connexió amb client remot), especificant la IP i el port on es vol connectar. CONNECT fa una apertura activa, enviant a l'altre host un paquet que té el bit SYN activat (el primer paquet que s'envia en el diagrama) i on s'indica el nombre de la seqüència inicial 0 que farà servir per tal d'enviar missatges.

El host receptor rep el segment i revisa si hi ha algun procés actiu que hagi executat un LISTEN en el port sol·licitat. Si existeix, el procés que rep el missatge rep el segment TCP, registra el número de seqüència 0, i respon amb un ACK 1 (núm. seq. + 1) i amb el bit SYN activat. A més, inclou el seu número de seqüència inicial 0 deixant oberta la connexió al seu extrem. Sinó hagués volgut establir la connexió hagués enviat una resposta amb el bit RST activat.

El nostre host rep la confirmació i a partir d'aquest moment pot enviar totes les dades des del seu extrem i viceversa.

En el quart segment veiem que ens ha arribat el Daytime, protocol ja vist al principi de l'exercici. En aquest segment s'activa el bit PSH el qual farà que totes les dades que s'hagin guardat en el nostre buffer seran enviades a l'aplicació.

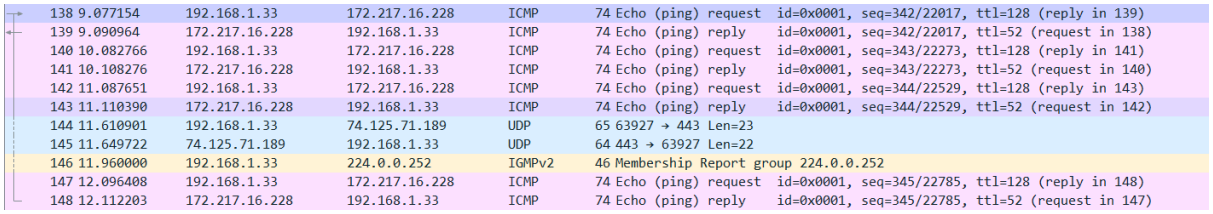
Una vegada el host del Daytime no té més dades a transmetre, envia un segment TCP amb el FIN activat i tanca el sentit d'enviament en la connexió, mantenint obert el sentit de recepció. El nostre host rep el segment amb el bit FIN, retorna la confirmació e informa a l'aplicació de la petició de tancament. És aquesta la que decideix el tancament de l'altre sentit de la connexió.

El nostre hoste tampoc té més dades a transmetre i envia un segment TCP amb els mateixos flags. Quan l'altre host rep el segment retorna l'ACK (tot i que ha tancat el seu sentit de connexió pot i ha d'enviar confirmacions). I finalment, quan el nostre hoste rep el segment ACK tanca definitivament la connexió.

## Exercici 3

**Des de capture, cliqueu a start i feu un ping a una adreça pública coneguda d'Internet (per exemple www.google.com). Captureu el transit de la comunicació i desglosseu la comunicació en sí. Expliqueu detalladament la captura, tal i com s'ha fet en l'exercici anterior. Què és el protocol ICMP? Com funciona aquest protocol? Quins identificadors i flags fa servir?**

La captura del sniffer ha sigut la següent:



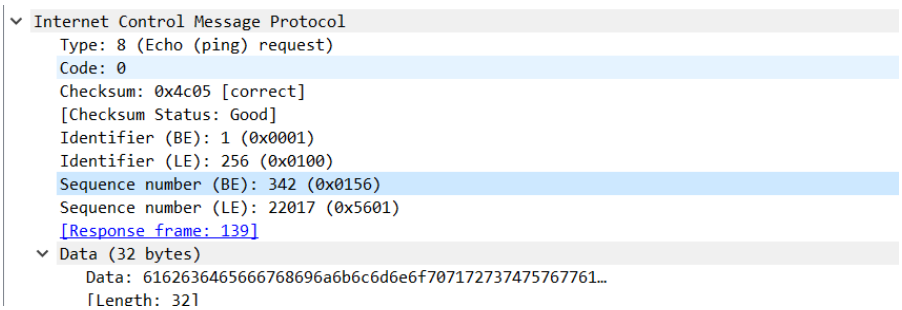
| | | | | | |
|---|---|---|---|---|---|
| 138 | 9.077154 | 192.168.1.33 | 172.217.16.228 | ICMP | 74 Echo (ping) request id=0x0001, seq=342/22017, ttl=128 (reply in 139) |
| 139 | 9.090964 | 172.217.16.228 | 192.168.1.33 | ICMP | 74 Echo (ping) reply id=0x0001, seq=342/22017, ttl=52 (request in 138) |
| 140 | 10.082766 | 192.168.1.33 | 172.217.16.228 | ICMP | 74 Echo (ping) request id=0x0001, seq=343/22273, ttl=128 (reply in 141) |
| 141 | 10.108276 | 172.217.16.228 | 192.168.1.33 | ICMP | 74 Echo (ping) reply id=0x0001, seq=343/22273, ttl=52 (request in 140) |
| 142 | 11.087651 | 192.168.1.33 | 172.217.16.228 | ICMP | 74 Echo (ping) request id=0x0001, seq=344/22529, ttl=128 (reply in 143) |
| 143 | 11.110390 | 172.217.16.228 | 192.168.1.33 | ICMP | 74 Echo (ping) reply id=0x0001, seq=344/22529, ttl=52 (request in 142) |
| 144 | 11.610901 | 192.168.1.33 | 74.125.71.189 | UDP | 65 63927 → 443 Len=23 |
| 145 | 11.649722 | 74.125.71.189 | 192.168.1.33 | UDP | 64 443 → 63927 Len=22 |
| 146 | 11.960000 | 192.168.1.33 | 224.0.0.252 | IGMPv2 | 46 Membership Report group 224.0.0.252 |
| 147 | 12.096408 | 192.168.1.33 | 172.217.16.228 | ICMP | 74 Echo (ping) request id=0x0001, seq=345/22785, ttl=128 (reply in 148) |
| 148 | 12.112203 | 172.217.16.228 | 192.168.1.33 | ICMP | 74 Echo (ping) reply id=0x0001, seq=345/22785, ttl=52 (request in 147) |

*Imatge 15. Captura de Wireshark amb paquets enviats i rebuts a partir de la comanda ping.*

Com podem veure i ja sabíem la comanda ping realitza 4 proves per defecte, sense aplicar cap altre condició a la comanda, i ens mostra els paquets rebuts, els perduts y la mitja del temps que ha trigat la comunicació en mil·lisegons.
Com s'envien llavors quatre paquets podem observar que s'han enviat i rebut quatre paquets, ja que els nostre ordinador forma els paquets "request" i el servidor de google ens contesta amb paquets "reply" per a cada una de les vegades que hem sol·licitat resposta.

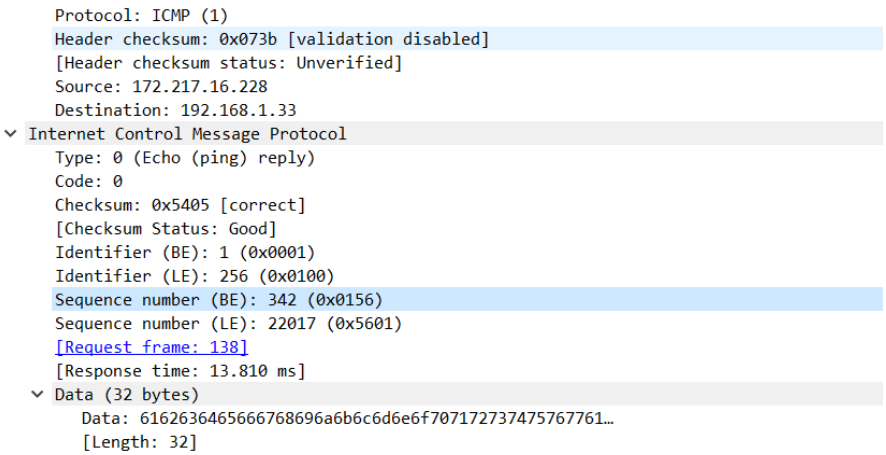
El protocol ICMP o "Internet Control Message Protocol" forma part del conjunt de protocols IP, i s'utilitza per enviar missatges d'error i informació operativa i no es normalment utilitzat per aplicacions d'usuari però ping o traceroute son una excepció que ens permeten utilitzar-lo per diagnòstics. Tot i això, aquest protocol pot ser també utilitzat per exemple quan un host no es pot trobar o el TTL ha expirat.

La trama ICMP ve definida principalment pel tipus i el codi, on el tipus defineix en la majoria dels casos de que es tracta la trama i defineix l'estructura que tindrà la resta, i el codi s'utilitza a vegades per poder especificar per exemple en quin moment s'ha trobat un error, de quin tipus és, etc.
Sempre després del codi tenim n "checksum" per comprovar que la trama es correcta.



```
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x4c05 [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence number (BE): 342 (0x0156)
    Sequence number (LE): 22017 (0x5601)
    [Response frame: 139]
  Data (32 bytes)
      Data: 6162636465666768696a6b6c6d6e6f707172737475767761…
      [Length: 32]
```

*Imatge 16. Trama "request" de ping www.google.com*



```
    Protocol: ICMP (1)
    Header checksum: 0x073b [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.217.16.228
    Destination: 192.168.1.33
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x5405 [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence number (BE): 342 (0x0156)
    Sequence number (LE): 22017 (0x5601)
    [Request frame: 138]
    [Response time: 13.810 ms]
  Data (32 bytes)
      Data: 6162636465666768696a6b6c6d6e6f707172737475767761…
      [Length: 32]
```

*Imatge 17. Trama "reply" per part del servidor de Google*

En el nostre cas el tipus de la trama ICMP son el 8 i 0, amb codi 0 ja que no hem d'especificar més que s'identifiquen com una rama de "request" i "reply".
El "request", després de la informació obligatòria per a cada trama ICMP, proporciona un identificador i número de seqüencia (que s'envien tant en Big Endian com en Little Endian) que s'utilitza per relacionar la trama "request" amb "reply" i per últim dades, que no han de tenir un sentit real, però que a la trama "reply" s'han de trobar de forma idèntica.

No hi ha una gran diferencia llavors a la trama "reply", funcionen de la mateixa forma i gràcies a que coneixem el temps en el que ha rebut el "request" i el que rebem la trama "reply", i també la relació entre "request" i "reply", podem saber el temps total des de que hem enviat el "request" fins que ens arriba el "reply" i els temps parcials que ha trigat el primer paquet i la contestació.

**Obriu el navegador i poseu http://ip_obtinguda a través del ping. S'obre la pàgina? Que captura el sniffer? Feu una explicació detallada.**
**Desglosseu la captura per connectar amb la web. Aneu al protocol TCP. Quin port de sortida heu fet servir? Identifiqueu algun protocol de control de flux? Expliqueu detalladament el que heu capturat.**

Si, la pagina s'obre i el sniffer captura els paquets que s'envien entre el servidor i el nostre ordinador per poder carregar-la, a demés de algunes comunicacions prèvies DNS, la informació en TCP i alguns paquets UDP, i HTTP.

| | | | | | | |
|---|---|---|---|---|---|---|
| 19 | 1.713273 | 192.168.1.104 | 93.184.220.29 | TCP | 55 | 58963 → 80 [ACK] Seq=1 Ack=1 Win=1023 Len=1 |
| 20 | 1.719980 | 192.168.1.104 | 172.217.17.10 | TCP | 66 | 58967 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 21 | 1.722318 | 192.168.1.104 | 172.217.17.10 | TCP | 66 | 58968 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 22 | 1.727200 | 192.168.1.104 | 192.168.1.1 | DNS | 74 | Standard query 0x3292 A www.google.com |
| 23 | 1.727819 | 192.168.1.1 | 192.168.1.104 | DNS | 90 | Standard query response 0x3292 A www.google.com A 172.217.168.164 |
| 24 | 1.729430 | 192.168.1.104 | 192.168.1.1 | DNS | 74 | Standard query 0x8907 AAAA www.google.com |
| 25 | 1.733998 | 172.217.17.10 | 192.168.1.104 | TCP | 66 | 80 → 58967 [SYN, ACK] Seq=0 Ack=1 Win=60720 Len=0 MSS=1380 SACK_PERM=1 WS=256 |
| 26 | 1.734077 | 192.168.1.104 | 172.217.17.10 | TCP | 54 | 58967 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0 |
| 27 | 1.735804 | 172.217.17.10 | 192.168.1.104 | TCP | 66 | 80 → 58968 [SYN, ACK] Seq=0 Ack=1 Win=60720 Len=0 MSS=1380 SACK_PERM=1 WS=256 |
| 28 | 1.735882 | 192.168.1.104 | 172.217.17.10 | TCP | 54 | 58968 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0 |
| 29 | 1.736320 | 93.184.220.29 | 192.168.1.104 | TCP | 66 | 80 → 58963 [ACK] Seq=1 Ack=2 Win=292 Len=0 SLE=1 SRE=2 |
| 30 | 1.744418 | 192.168.1.104 | 172.217.168.164 | TCP | 66 | 58969 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 31 | 1.744421 | 192.168.1.1 | 192.168.1.104 | DNS | 102 | Standard query response 0x8907 AAAA www.google.com AAAA 2a00:1450:4003:803::2004 |
| 32 | 1.756161 | 172.217.168.164 | 192.168.1.104 | TCP | 66 | 443 → 58969 [SYN, ACK] Seq=0 Ack=1 Win=60720 Len=0 MSS=1380 SACK_PERM=1 WS=256 |
| 33 | 1.756291 | 192.168.1.104 | 172.217.168.164 | TCP | 54 | 58969 → 443 [ACK] Seq=1 Ack=1 Win=262144 Len=0 |
| 34 | 1.757805 | 192.168.1.104 | 172.217.168.164 | TLSv1.3 | 571 | Client Hello |
| 35 | 1.769480 | 172.217.168.164 | 192.168.1.104 | TCP | 60 | 443 → 58969 [ACK] Seq=1 Ack=518 Win=61952 Len=0 |
| 36 | 1.790058 | 172.217.168.164 | 192.168.1.104 | TLSv1.3 | 1484 | Server Hello, Change Cipher Spec |
| 37 | 1.790059 | 172.217.168.164 | 192.168.1.104 | TLSv1.3 | 1249 | Application Data |
| 38 | 1.790116 | 192.168.1.104 | 172.217.168.164 | TCP | 54 | 58969 → 443 [ACK] Seq=518 Ack=2626 Win=262144 Len=0 |
| 39 | 1.793979 | 192.168.1.104 | 192.168.1.1 | DNS | 73 | Standard query 0x0dd9 A ocsp.pki.goog |
| 40 | 1.807991 | 192.168.1.1 | 192.168.1.104 | DNS | 124 | Standard query response 0x0dd9 A ocsp.pki.goog CNAME pki-goog.l.google.com A 172.217.16.227 |
| 41 | 1.808759 | 192.168.1.104 | 172.217.16.227 | TCP | 66 | 58970 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 42 | 1.808897 | 192.168.1.104 | 192.168.1.1 | DNS | 81 | Standard query 0xc118 A pki-goog.l.google.com |
| 43 | 1.810791 | 192.168.1.1 | 192.168.1.104 | DNS | 97 | Standard query response 0xc118 A pki-goog.l.google.com A 172.217.16.227 |
| 44 | 1.811158 | 192.168.1.104 | 192.168.1.1 | DNS | 81 | Standard query 0x73a1 AAAA pki-goog.l.google.com |
| 45 | 1.820634 | 172.217.16.227 | 192.168.1.104 | TCP | 66 | 80 → 58970 [SYN, ACK] Seq=0 Ack=1 Win=60720 Len=0 MSS=1380 SACK_PERM=1 WS=256 |
| 46 | 1.820716 | 192.168.1.104 | 172.217.16.227 | TCP | 54 | 58970 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0 |
| 47 | 1.820939 | 192.168.1.104 | 172.217.16.227 | OCSP | 459 | Request |
| 48 | 1.826910 | 192.168.1.1 | 192.168.1.104 | DNS | 109 | Standard query response 0x73a1 AAAA pki-goog.l.google.com AAAA 2a00:1450:4003:80b::2003 |
| 49 | 1.831987 | 172.217.16.227 | 192.168.1.104 | TCP | 60 | 80 → 58970 [ACK] Seq=1 Ack=406 Win=61952 Len=0 |

*Imatge 18. Comunicacions quan accedim a http://172.217.17.10 pel navegador*

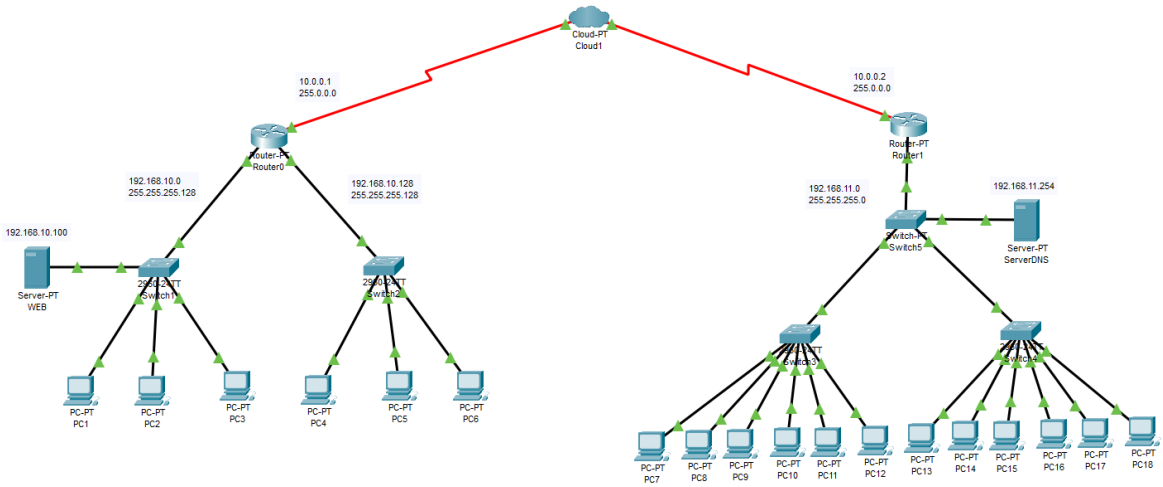
Encara que no tenim per segur que sigui així, hem observat que el primer cop que hem provat a buscar http://172.217.17.10 semblava que havia hagut una connexió DNS amb una query y una resposta on es preguntava la ip de www.google.com, però sembla que amb les proves següents el resultat no ha sigut el mateix i no ha sigut necessari preguntar quina es aquesta direcció, hem repetit el procediment múltiples vegades i hem arribat a la conclusió de que el protocol DNS no s'està utilitzant moltes vegades ja que el ordinador recorda quina es la direcció a la web. Inicialment pensàvem que no utilitzaria DNS en cap moment ja que no

seria necessari si ja sabem la direcció IP del servidor però sembla que igualment hem de traduir la IPv4 a IPv6 i s'utilitza igualment el protocol DNS per obtenir-la, a més sempre s'utilitza per informació extra de la web o relacionat, una d'elles es www.gstatic.com.
La informació s'envia a través del protocol TCP i per començar es detecta que tant el client com el servidor es fan un "Hello" (Handshake Protocol) per saber que hi ha connexió entre ells i que s'han d'enviar informació.
Com hem vist a teoria el servidor i el client s'envien informació entre si a través dels paquets TCP (podem veure també TLSv1.3, que conté les dades de la aplicació), aquests paquets s'utilitzen per controlar que la informació estigui arribant correctament entre els dos i podem veure com inicialment s'envia SYN per poder sincronitzar les comunicacions per part del usuari, el servidor contesta amb el SYN, ACK, es produeix el Handshake Protocol, i el servidor i el client s'envien dades, i els dos fan el ACK de les dades que els hi arriba fins que, en el nostre cas el servidor acaba fent el ACK final.
Els ports que s'han utilitzat són el d'origen 58969 i  destí 443.

## Exercici 4

**Obre el programa simulador Cisco Packet Tracer i crea dues xarxes connectades a través del núvol. Cada una de les xarxes ha de tenir un router genèric i dos switchos de 24 canals. Connecta uns quants ordinadors (entre cinc i vuit) als switchos. En una de les xarxes munta un subnetting variable i assigna les IP's de manera estàtica. A l'altra xarxa munta en el router un servidor DHCP per tal d'assignar IPs de forma dinàmica. En totes dues xarxes configura NAT.**

L'esquema de la xarxa que hem configurat es la següent:



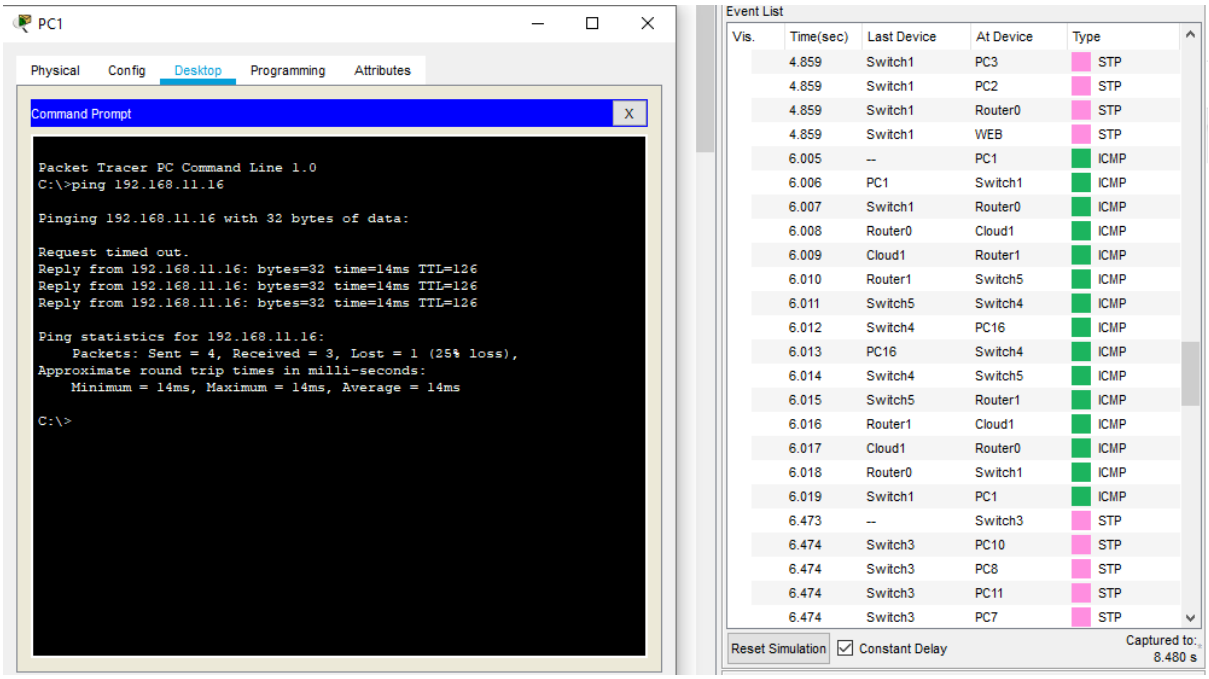

*Imatge 19. Xarxa demanada*

No hem pogut configurar NAT ja que la configuració per terminal dels Routers no ens deixaven accés a la configuració o algun problema relacionat amb aquest pas, però hem configurat rutes estàtiques entre les dues xarxes per a que es puguin comunicar entre si.
Com es pot veure a la dreta hem creat la xarxa on hem configurat dues sortides del Router com a subxarxes que van a cada un dels "switch" i a cada un hem connectat tres ordinadors i a més al primer d'ells un Servidor per que faci Host de la pagina web, a la qual podem accedir des de qualsevol altre lloc de la xarxa. Hem dividit aquesta xarxa estàtica en dues amb

mascara 255.255.255.128, per tant deixant la primera meitat tenint lloc per connectar fins a 126 dispositius entre la IP 192.168.10.1 fins a la 192.168.10.127, i la segona la resta entre la 192.168.10.130 a la 192.168.10.255. El router ("Host") es coneix amb la IP 192.168.10.1 a la primera subxarxa i amb 192.168.10.129 a la segona.
Hem configurat llavors les IP estàtiques dels ordinadors personals que van de la 192.168.10.10 a la 192.168.10.12 i de la 192.168.10.139 a la 192.168.10.141.
A la segona xarxa per hem pensat que la millor forma de crear la xarxa amb IP dinàmiques i poder tenir connectat un servidor DNS de la forma més fàcil es connectar els dos "switch" a un altre ja que les connexions directes al "router" s'entenen com a diferents subxarxes, així hem configurat el servei de DNS que ens proporciona 100 adreces a partir de la IP 192.168.11.15 (mascara 255.255.255.0). A partir d'aquí podem connectar PC's i automàticament, dient que utilitzin DHCP, reben una IP i no es necessari configurar totes les adreces manualment.

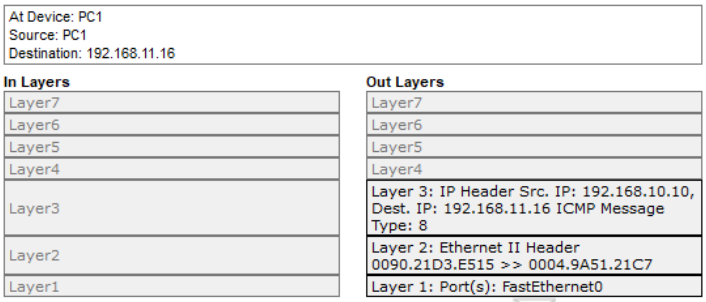
**i) Fes PING des de un dels ordinadors de la xarxa dinàmica a un dels ordinadors de l'altra xarxa. Mostra els diferents empaquetats que es realitzen.**

Quan fem ping si el NAT estigues configurat podríem veure com la IP de origen canvia en el moment que el paquet travessa el Router, i es traduiria la IP privada, per exemple 192.168.10.10, a una IP pública, probablement amb forma 10.0.0.1.
El primer ping que hem fet ha tingut un "Time Out" ja que el protocol STP estava reconeixent les connexions entre dispositius però a partir d'aquí tot ha funcionar correctament i aquets son els resultats



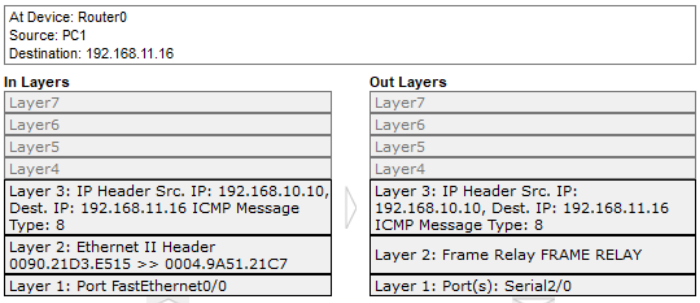

*Imatge 20. Ping del ordinador PC1 al PC16 a Packet Tracer*

Es pot observar que el paquet viatja del PC1 al PC16 i torna, durant aquest camí es interesant veure la trama que hi ha inicialment , al final i als "Routers".

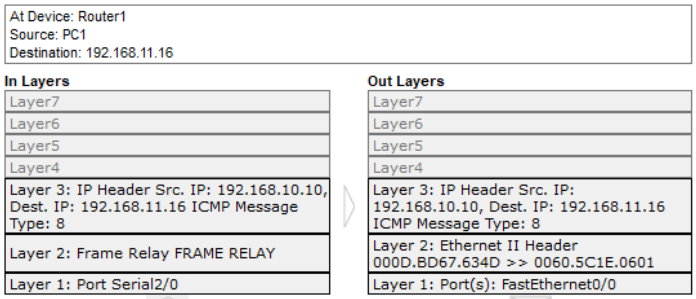



*Imatge 21. Trama "request" a l'ordinador PC1*

Podem veure clarament com a l'origen s'envia la trama per FastEthernet0 amb "source" 192.168.10.10 i "Destination" 192.168.11.16

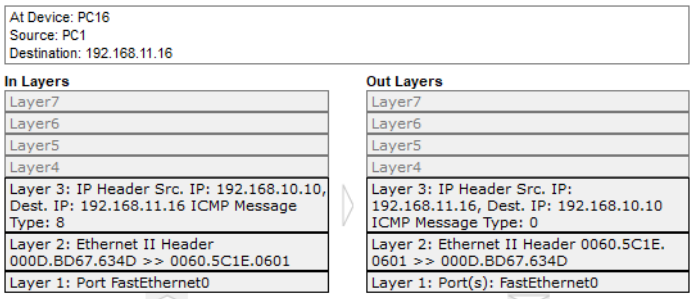



*Imatge 22. Trama "request" al Router0*




*Imatge 23. Trama "request" al Router1*

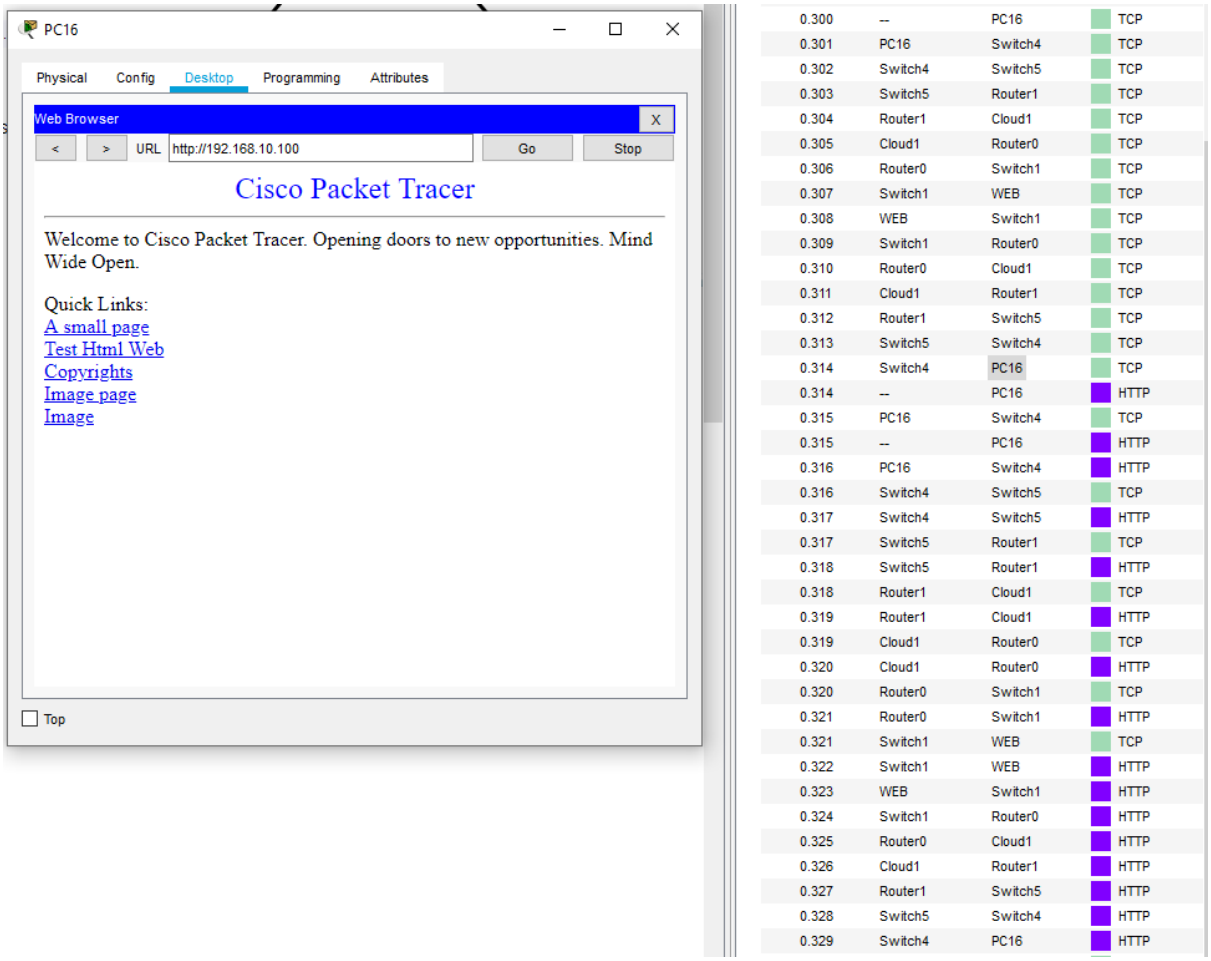
Podem veure com a la sortida de la primera xarxa i la entrada de la segona la connexió utilitzada es modificada i en comptes de ethernet s'utilitza Frame Relay a través de Serial




*Imatge 24. Trama "request" arriba al PC16 i trama de "reply" que genera*

En arribar al destí aquest crea la trama "reply" i s'envia al origen de la trama "request", es a dir ara el destí es l'ordinador des de el qual hem fet el ping i el paquet segueix el camí invers per arribar al nou destí.

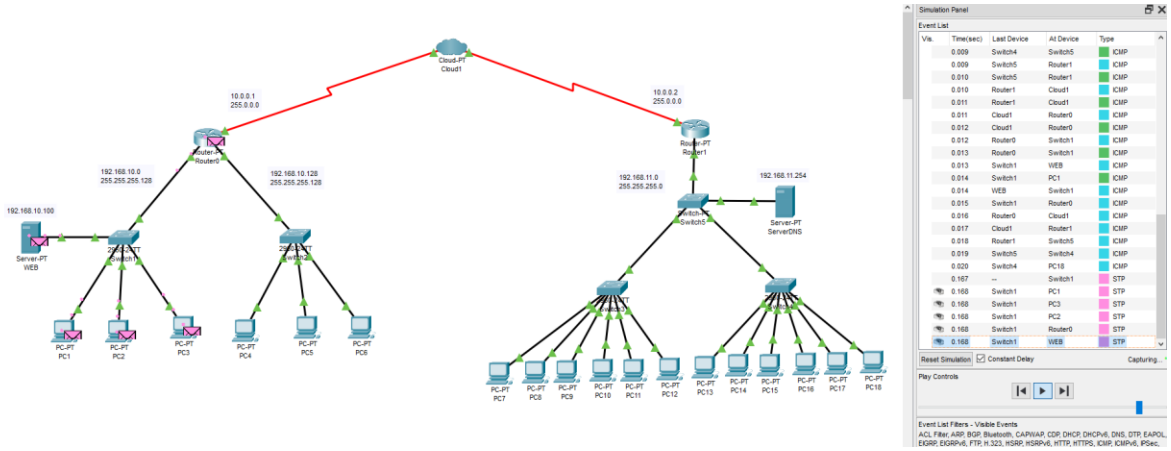
**ii) Configura una pàgina web en un dels equips i connectat des de l'altra xarxa a aquesta pàgina. Mostra els diferents empaquetats que es realitzen.**




*Imatge 25. Seqüencia per carregar la pàgina web al ordinador PC16*

Aquestes son les comunicacions entre servidor i usuari que s'han produït per poder carregar la pagina web des de el PC16, com podem veure la primera part de la comunicació es el "Handshake Protocol" on tant servidor com client es posen d'acord en establir la comunicació i les seves característiques, en segon lloc el PC16 fa el "request" de la informació de la pàgina web i l'envia amb encapsulació HTTP al servidor i aquest contesta amb la informació amb el mateix format.

**iii) Mostra una imatge on es visualitzi la xarxa. Captura el tràfic generat.**



*Imatge 26. Xarxa amb tràfic comú*

## CONCLUSIONS

En conclusió, creiem que hem après moltes coses noves realitzant aquesta pràctica i que ens hem familiaritzat bastant amb el programa Wireshark. Creiem que hem assolit els objectius d'aquesta pràctica i que hem après com treballen diferents protocols a l'hora de transmetre informació entre diversos equips. A més, hem profunditzat amb les capçaleres d'alguns protocols, i hem vist com funcionen els seus camps i la importància que tenen cadascun dels seus bits.