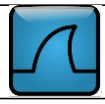
Étude de trames Ethernet à l'aide de Wireshark

Maintenant que nous avons pris en main des utilitaires de diagnostic réseau, nous allons observer les trames échangées dans le cadre des communications LAN/WAN.

L'outil utilisé sera un analyseur réseau : WireShark.	

1. Prise en main du logiciel WireShark.

Prendre connaissance du document `Fiche-Wireshark.pdf`.

Ouvrir le logiciel et suivre le tutoriel afin d'être capable de :

- Sélectionner une interface réseau à tracer,
- Lancer une capture sur cette interface,
- Arrêter une capture,
- Sélectionner un paquet (une trame) afin d'observer les couches protocolaires.

2. Capturer les trames lors d'un Ping

Il s'agit de préparer votre machine afin de capturer toutes les trames qui résultent de l'exécution de la commande ping.

Deux protocoles seront normalement engagés :

- ARP : Afin de d'obtenir l'adresse MAC du destinataire pour construire une trame Ethernet
- ICMP : Protocole utilisé par la commande PING

Trames qui résultent d'un PING :

4	4.098375000	HewlettP_95:22:54	Broadcast	ARP	42 Who has 192.168.2.17? Tell 192.168.2.18
5	4.099108000	HewlettP_95:22:b0	HewlettP_95:22:54	ARP	60 192.168.2.17 is at 2c:41:38:95:22:b0
6	4.099111000	192.168.2.18	192.168.2.17	ICMP	74 Echo (ping) request id=0x0200, seq=6400/25, ttl=128 (reply in 7)
7	4.100064000	192.168.2.17	192.168.2.18	ICMP	74 Echo (ping) reply id=0x0200, seq=6400/25, ttl=128 (request in 6)
8	5.088543000	192.168.2.18	192.168.2.17	ICMP	74 Echo (ping) request id=0x0200, seq=6656/26, ttl=128 (no response found!)
9	5.089291000	192.168.2.17	192.168.2.18	ICMP	74 Echo (ping) reply id=0x0200, seq=6656/26, ttl=128 (request in 8)
10	6.088535000	192.168.2.18	192.168.2.17	ICMP	74 Echo (ping) request id=0x0200, seq=6912/27, ttl=128 (reply in 11)
11	6.089279000	192.168.2.17	192.168.2.18	ICMP	74 Echo (ping) reply id=0x0200, seq=6912/27, ttl=128 (request in 10)
12	7.088520000	192.168.2.18	192.168.2.17	ICMP	74 Echo (ping) request id=0x0200, seq=7168/28, ttl=128 (reply in 13)
13	7.089268000	192.168.2.17	192.168.2.18	ICMP	74 Echo (ping) reply id=0x0200, seq=7168/28, ttl=128 (request in 12)

Voici ce qui résulte d'un PING à l'adresse 192.168.2.17 la mienne étant 192.168.2.18. Une demande est envoyée à l'adresse xxx.xxx.x.17 une réponse est reçue à l'adresse xxx.xxx.x.18. Ici ces deux actions se répètent 4 fois comme on peut le voir dans l'invite de commandes :

```
Réponse de 192.168.2.19 : octets=32 temps<1ms TTL=128
Réponse de 192.168.2.19 : octets=32 temps<1ms TTL=128
Réponse de 192.168.2.19 : octets=32 temps<1ms TTL=128
Réponse de 192.168.2.19 : octets=32 temps<1ms TTL=128
```

Une question est envoyée par mon poste:
qui à l'adresse « IP xxx.xxx.x.17 » ?? → je reçoit une réponse : c'est « Adresse MAC » qui à cette IP. Jmon poste commence donc à lui envoyer des paquets (request/reply)

2.1. Préparation

Vider le cache ARP (voir TP1) et s'assurer qu'il est vide. (Si vous n'êtes pas administrateur, pinguer un PC dont l'adresse MAC n'est pas encore présente dans le cache ARP).

Préparer un Ping sur un élément de votre choix.

2.2. Capture :

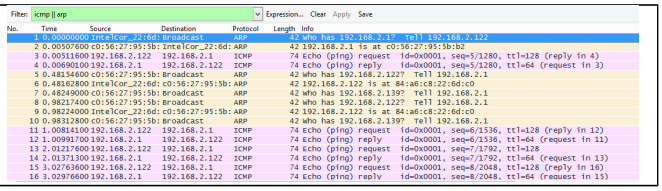
Capturer l'ensemble des trames du Ping. (Typiquement, 2 trames ARP et 8 trames ICMP)

Enregistrer la capture dans un fichier pour la dépouiller ultérieurement.

Filtrer uniquement les trames ARP et ICMP pour mieux analyser les trames.

A droite, allure de votre capture attendue :

Copier-coller votre capture CI-DESSOUS



4	4.098375000	HewlettP_95:22:54	Broadcast	ARP	42 Who has 192.168.2.17? Tell 192.168.2.18
5	4.099108000	HewlettP_95:22:b0	HewlettP_95:22:54	ARP	60 192.168.2.17 is at 2c:41:38:95:22:b0
6	4.099111000	192.168.2.18	192.168.2.17	ICMP	74 Echo (ping) request id=0x0200, seq=6400/25, ttl=128 (reply in 7)
7	4.100064000	192.168.2.17	192.168.2.18	ICMP	74 Echo (ping) reply id=0x0200, seq=6400/25, ttl=128 (request in 6)
8	5.088543000	192.168.2.18	192.168.2.17	ICMP	74 Echo (ping) request id=0x0200, seq=6656/26, ttl=128 (no response found!)
9	5.089291000	192.168.2.17	192.168.2.18	ICMP	74 Echo (ping) reply id=0x0200, seq=6656/26, ttl=128 (request in 8)
10	6.088535000	192.168.2.18	192.168.2.17	ICMP	74 Echo (ping) request id=0x0200, seq=6912/27, ttl=128 (reply in 11)
11	6.089279000	192.168.2.17	192.168.2.18	ICMP	74 Echo (ping) reply id=0x0200, seq=6912/27, ttl=128 (request in 10)
12	7.088520000	192.168.2.18	192.168.2.17	ICMP	74 Echo (ping) request id=0x0200, seq=7168/28, ttl=128 (reply in 13)
13	7.089268000	192.168.2.17	192.168.2.18	ICMP	74 Echo (ping) reply id=0x0200, seq=7168/28, ttl=128 (request in 12)

Puis commenter en référençant les N° de trames (colonne de gauche).

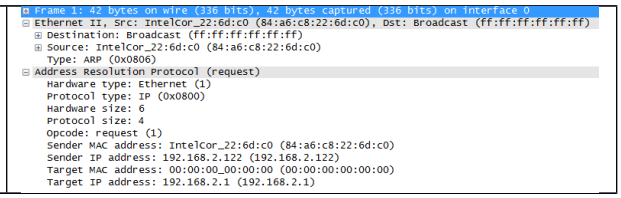
N° de Trame	Commentaire
4	Qui à 192.168.2.17 (ip)? dis 192.168.2.18(ip) //Paquet ARP
5	192.168.2.17(ip) est à 2C:41:38:95:22:b0(mac) //Paquet ARP
6	Début de la conversation :192.168.2.17 à une requete //Paquet ICMP
7	192.168.2.18 à une réponse //Paquet ICMP

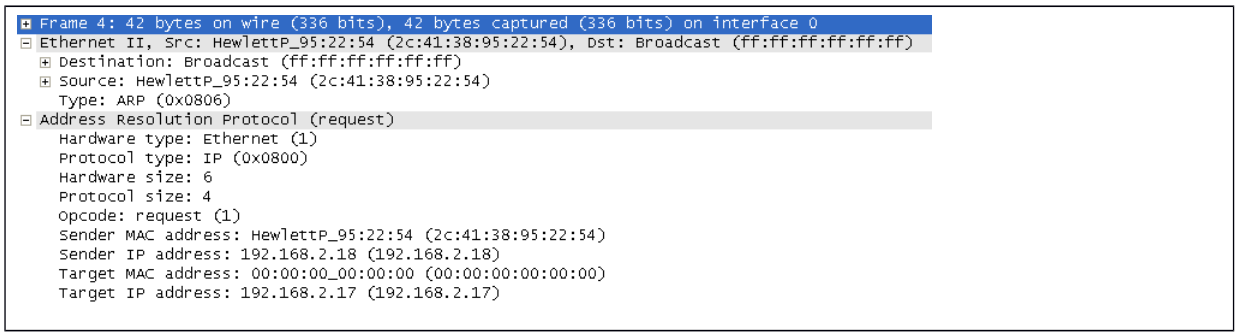
3. Encapsulation du paquet

3.1. Paquets ARP

3.1.1. Premier paquet

Reprendre la capture précédente et s'intéresser au tout premier paquet ARP (fenêtre centrale).

A droite, allure de votre capture attendue : Coller CI-DESSOUS une capture d'écran de ce paquet faisant apparaître les détails.	



Sur la couche Ethernet II, noter :

- les adresses MAC source et destination :

MAC source: 2C:41:38:95:22:54 MAC destination : 00:00:00:00:00:00

- le Type de trame encapsulé :

Trame de type ARP (0x0806)

Couche ARP notez le code fonction (Opcode):

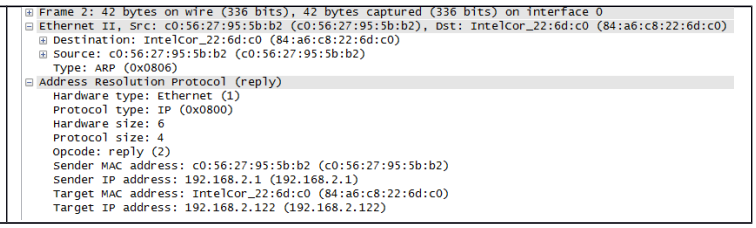
Opcode sert a dire quesque la question : ici c'est une demande (la 1ere)

Synthèse : A qui s'adresse ce message ? Que dit ce message ?

Ce premier message demande au réseau quelle « adresse mac » à « telle adresse IP »

3.1.2.Second paquet

Reprendre la capture précédente, et s'intéresser au second paquet ARP (fenêtre centrale).

A droite, allure de votre capture attendue : Coller CI-DESSOUS une capture d'écran de ce paquet faisant apparaître les détails.	

```
⊞ Frame 5: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
⊟ Ethernet II, Src: HewlettP_95:22:b0 (2c:41:38:95:22:b0), Dst: HewlettP_95:22:54 (2c:41:38:95:22:54)
  ⊞ Destination: HewlettP_95:22:54 (2c:41:38:95:22:54)
  ⊞ Source: HewlettP_95:22:b0 (2c:41:38:95:22:b0)
    Type: ARP (0x0806)
    Padding: 000000000000000000000000000000000000
⊟ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: HewlettP_95:22:b0 (2c:41:38:95:22:b0)
    Sender IP address: 192.168.2.17 (192.168.2.17)
    Target MAC address: HewlettP_95:22:54 (2c:41:38:95:22:54)
    Target IP address: 192.168.2.18 (192.168.2.18)
```

Noter le code fonction (Opcode couche ARP) :

Opcode : réponse (2)

Qu'indique la seconde trame ARP ?

Cette trame ARP est une réponse à la question envoyée auparavant, en effet nous obtenons une adresse MAC : 2c:41:38:95:22:b0

3.2. Paquet ICMP

Reprendre la démarche et étudier le paquet ICMP Echo request (ping).

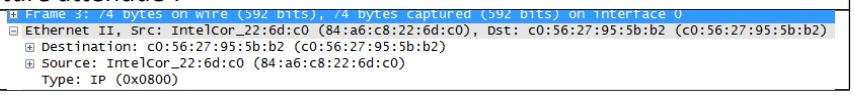
Cette fois, il y a 3 couches :

- Ethernet II,
- IP,
- ICMP.

Faire une capture d'écran des trois couches que vous commenterez ci-après.

3.2.1.Couche Ethernet II

Allure de votre capture attendue :



Coller CI-DESSOUS une capture d'écran de ce paquet faisant apparaître les détails.

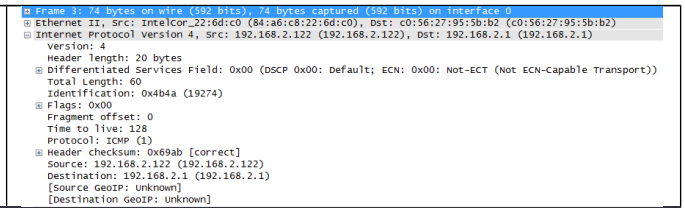
```
⊞ Frame 6: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
⊟ Ethernet II, Src: HewlettP_95:22:54 (2c:41:38:95:22:54), Dst: HewlettP_95:22:b0 (2c:41:38:95:22:b0)
  ⊞ Destination: HewlettP_95:22:b0 (2c:41:38:95:22:b0)
  ⊞ Source: HewlettP_95:22:54 (2c:41:38:95:22:54)
    Type: IP (0x0800)
```

Commentez les informations contenues :

Nous avons ici, l'adresse MAC de la destination du paquet, mais aussi l'adresse MAC de sa source (son « envoyeur »)

3.2.2.Couche IP

La couche IP (Internet Protocole) se charge de faire voyager une trame à travers un ou plusieurs réseaux en empruntant des routeurs.

A droite, allure de votre capture attendue :	

Coller CI-DESSOUS une capture d'écran de ce paquet faisant apparaître les détails.

```
Frame 6: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
Ethernet II, Src: HewlettP_95:22:54 (2c:41:38:95:22:54), Dst: HewlettP_95:22:b0 (2c:41:38:95:22:b0)
Internet Protocol Version 4, Src: 192.168.2.18 (192.168.2.18), Dst: 192.168.2.17 (192.168.2.17)
    Version: 4
    Header Length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 60
    Identification: 0x8bd3 (35795)
  Flags: 0x00
    Fragment offset: 0
    Time to live: 128
    Protocol: ICMP (1)
  Header checksum: 0x0000 [validation disabled]
    Source: 192.168.2.18 (192.168.2.18)
    Destination: 192.168.2.17 (192.168.2.17)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
Internet Control Message Protocol
```

A savoir
L a couche IP (Internet Protocole) référence les adresses IPs logiques **source** et **destination**
Elle fragmente éventuellement le message transporté à l'émission et le et réassemble à la réception.
Elle applique un temps de vie, vérifie sa conformité (checksum), précise le type de protocole quelle transporte.

Trouver et relevez les informations pertinentes de cette couche :

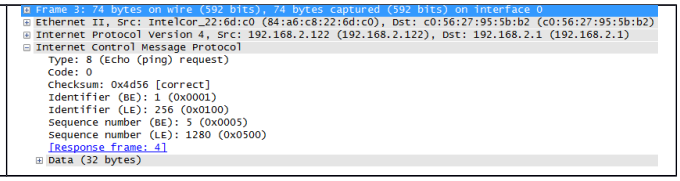
Protocole : ICMP Source : 192. 168.2.18 Destination : 192. 168.2.17

Sélectionner, sur Wireshark, un champ **adresse IP** (source ou destination) afin de visualiser les octets concernés dans le cadre du bas. Justifier les valeurs obtenues. En déduire le nombre d'octets nécessaire pour coder une adresse IPV4 ainsi que l'étendue des valeurs possibles.

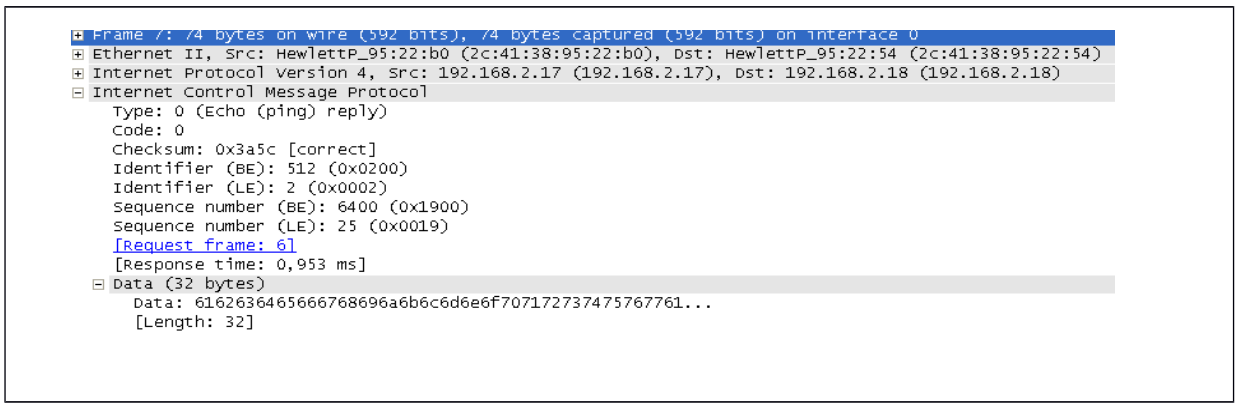
```
0000  2c 41 38 95 22 b0 2c 41  38 95 22 54 08 00 45 00   ,A8."·,A 8."T··E·
0010  00 3c 8b d3 00 00 80 01  00 00 c0 a8 02 12 c0 a8   ·<·····  ········
0020  02 11 08 00 32 5c 02 00  19 00 61 62 63 64 65 66   ····2\·· ··abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                    wabcdefg hi
```

Codé sur 8 octets, soit 255 valeurs d'IP possibles.

3.2.3.Couche ICMP

A droite, allure de votre capture attendue :	

Coller CI-DESSOUS une capture d'écran de ce paquet faisant apparaître les détails.



Commentaires : Quelle information principale porte cette couche ?

l'information principale que porte cette couche est le **le message de contrôle d'erreur (Checksum)**

Observer et comparer les types ICMP (couche ICMP) des trames aller et retour du ping.

Trames aller du ping : Type request (demande) Trames retour du ping : Type reply (réponse)

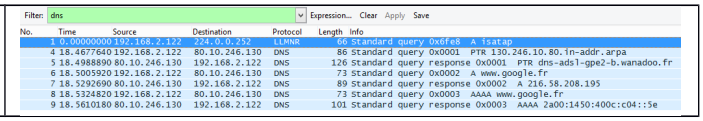
4. Trames requête DNS

Reprendre la même démarche pour observer une requête DNS. Vous utiliserez la commande `nslookup` (vu dans le TP1) afin de comparer son résultat et les informations contenues dans les captures réseau.

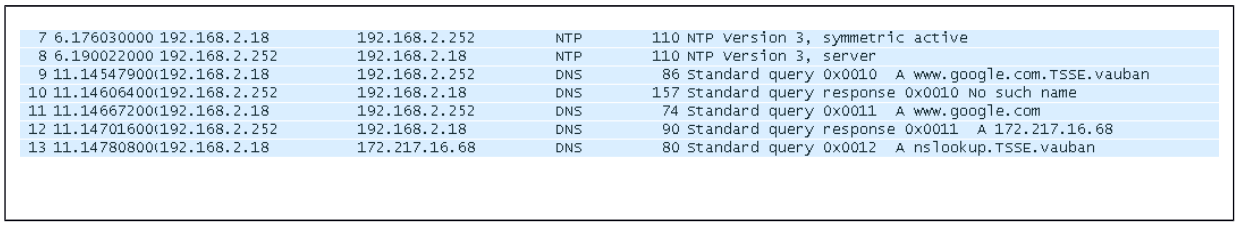
Vous ne vous intéresserez qu'aux couches IP, UDP et DNS.

A savoir
Le paquet DNS est encapsulé dans une trame UDP/IP c'est-à-dire UDP transporté par IP.
UDP est un mode de communication dit « non connecté » c.-à-d. qui ne garantit pas le fait que le destinataire reçoive le message (contrairement à TCP dit mode connecté).
Son rôle essentiel est de définir le service cible du message à travers un port (53 pour DNS)

4.1. Identifier les paquets des requêtes et des réponses.

A droite, allure de votre capture attendue : Copier-coller la capture CI-DESSOUS	

Faire le nécessaire pour générer une requête DNS



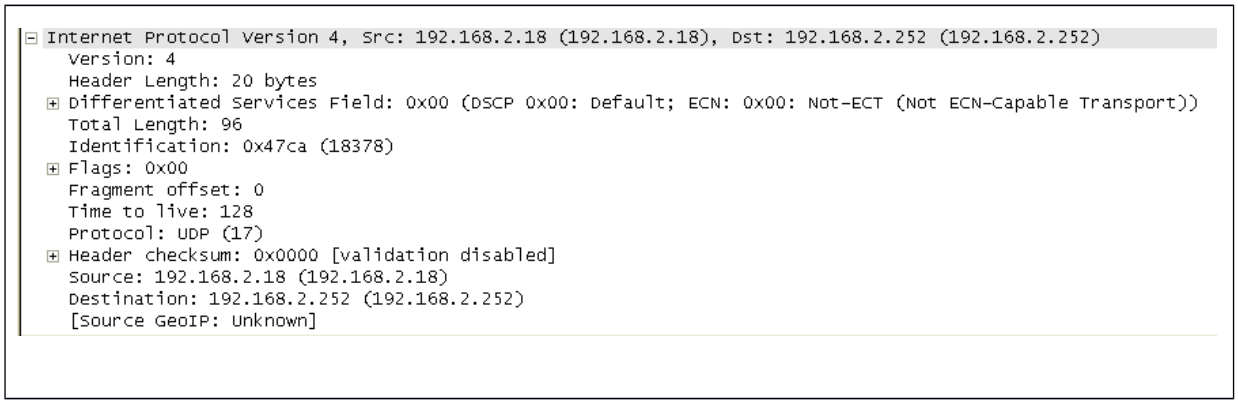
```
  7 6.176030000 192.168.2.18    192.168.2.252   NTP      110 NTP Version 3, symmetric active
  8 6.190022000 192.168.2.252   192.168.2.18    NTP      110 NTP Version 3, server
  9 11.14547900(192.168.2.18    192.168.2.252   DNS       86 Standard query 0x0010  A www.google.com.TSSE.vauban
 10 11.14606400(192.168.2.252   192.168.2.18    DNS      157 Standard query response 0x0010 No such name
 11 11.14667200(192.168.2.18    192.168.2.252   DNS       74 Standard query 0x0011  A www.google.com
 12 11.14701600(192.168.2.252   192.168.2.18    DNS       90 Standard query response 0x0011  A 172.217.16.68
 13 11.14780800(192.168.2.18    172.217.16.68   DNS       80 Standard query 0x0012  A nslookup.TSSE.vauban
```

Compléter le tableau ci-dessous :

N°de paquet	Commentaire
7	Mon PC dit au serveur qu'il vas lancer une requête DNS
8	Le serveur dit qu'il est prêt et qu'il a reçu l'information
9	Le PC envoie un « questionnement standart » à www.google.com.TSSE.vauban
10	Le serveur reçoit l'information, répond « pas de tel nom »
11	Le PC envoie un « questionnement standart » à www.google.com
12	Le serveur reçoit l'information, répond «vas va à l'adresse 172.217.16.68» (si on tape cette adresse dans la barre de recherche d'un navigateur, nous arriverons sur www.google.com)

4.2. Couche IP

Capture :



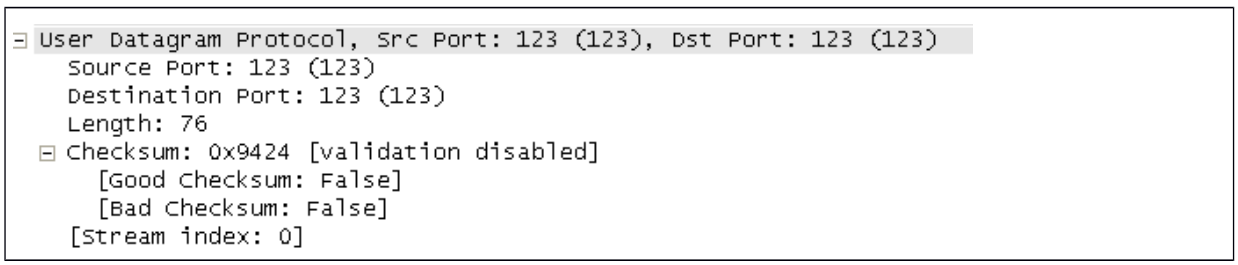
```
⊟ Internet Protocol Version 4, Src: 192.168.2.18 (192.168.2.18), Dst: 192.168.2.252 (192.168.2.252)
    Version: 4
    Header Length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 96
    Identification: 0x47ca (18378)
  ⊞ Flags: 0x00
    Fragment offset: 0
    Time to live: 128
    Protocol: UDP (17)
  ⊞ Header checksum: 0x0000 [validation disabled]
    Source: 192.168.2.18 (192.168.2.18)
    Destination: 192.168.2.252 (192.168.2.252)
    [Source GeoIP: Unknown]
```

Dans ces paquets, retrouver les informations principales :

Source du paquet
Destination du paquet
Valeur du checksum

4.3. Couche UDP

Capture



```
⊟ User Datagram Protocol, Src Port: 123 (123), Dst Port: 123 (123)
    Source Port: 123 (123)
    Destination Port: 123 (123)
    Length: 76
  ⊟ Checksum: 0x9424 [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
    [Stream index: 0]
```

Dans ces paquets, retrouver les informations principales :

Port source
Port de destination
Taille du message
Checksum

4.4. Couche DNS

Capture

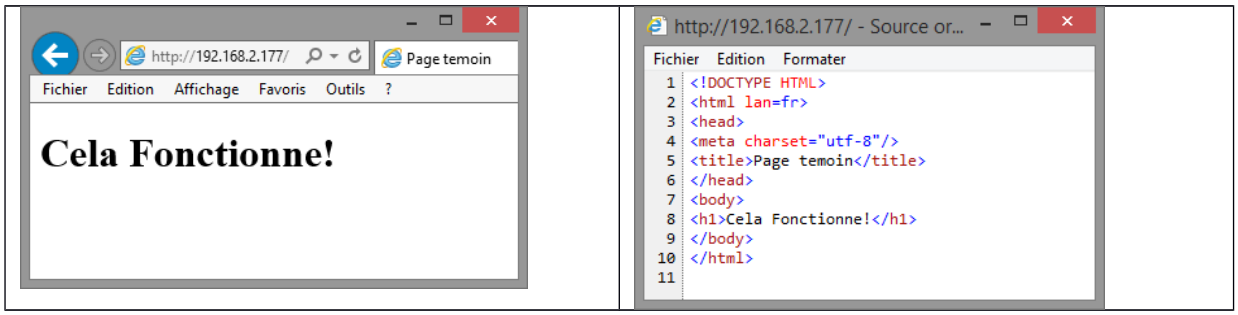
```
Domain Name System (query)
    [Response In: 10]
    Transaction ID: 0x0010
  Flags: 0x0100 Standard query
      0... .... .... .... = Response: Message is a query
      .000 0... .... .... = Opcode: Standard query (0)
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... .0.. .... = Z: reserved (0)
      .... .... ...0 .... = Non-authenticated data: Unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  Queries
```

Dans ces paquets, retrouver les informations principales :

Temps de réponse
nombre de questions
de réponses

5. Trame requête http

Dans cette partie, il s'agit de d'observer les trames échangées entre un navigateur web et un serveur Web.



Le serveur Web aura l'adresse 192.168.2.177

A l'aide d'un navigateur, demander la page web ci-dessus et capturer les rames avec Wireshark.

Ci-dessous les trois trames lors de l'établissement de la connexion

```
16 4.335590000 192.168.2.18    192.168.2.177   TCP    62 2931→80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
17 4.335838000 192.168.2.177   192.168.2.18    TCP    60 80→2931 [SYN, ACK] Seq=0 Ack=1 Win=2048 Len=0 MSS=1460
18 4.335856000 192.168.2.18    192.168.2.177   TCP    54 2931→80 [ACK] Seq=1 Ack=1 Win=65535 Len=0
```

Le protocole TCP utilise trois trames pour réussir une connexion :
[SYNC] demande de connexion client->serveur
[SYNC-ACK] Acceptation de la demande de connexion serveur -> client
[ACK] Acquittement Client->Serveur (le client confirme la connexion TCP, les requêtes http vont pouvoir commencer/

Ci-dessous la demande de page web du client

```
19 4.341538000 192.168.2.18    192.168.2.177   HTTP   355 GET / HTTP/1.1
```

Protocole http le client demande une page web avec un GET<chemin> au serveur.
Le serveur répond en envoyant des Paquets TCP contenant, entre autre, le code HTML

Ci- dessous la réponse du serveur avec les fragmentations de paquets TCP.

```
21 4.544310000 192.168.2.177   192.168.2.18    TCP     60 80→2931 [ACK] Seq=1 Ack=302 Win=1994 Len=0
22 4.601783000 192.168.2.177   192.168.2.18    TCP    132 [TCP segment of a reassembled PDU]
23 4.601817000 192.168.2.177   192.168.2.18    TCP     60 [TCP segment of a reassembled PDU]
24 4.601828000 192.168.2.18    192.168.2.177   TCP     54 2931→80 [ACK] Seq=302 Ack=81 Win=65455 Len=0
25 4.602184000 192.168.2.177   192.168.2.18    TCP     60 [TCP segment of a reassembled PDU]
26 4.604199000 192.168.2.177   192.168.2.18    TCP    166 [TCP segment of a reassembled PDU]
```

Ci-dessous, localiser la trame qui porte le code HTML de la page

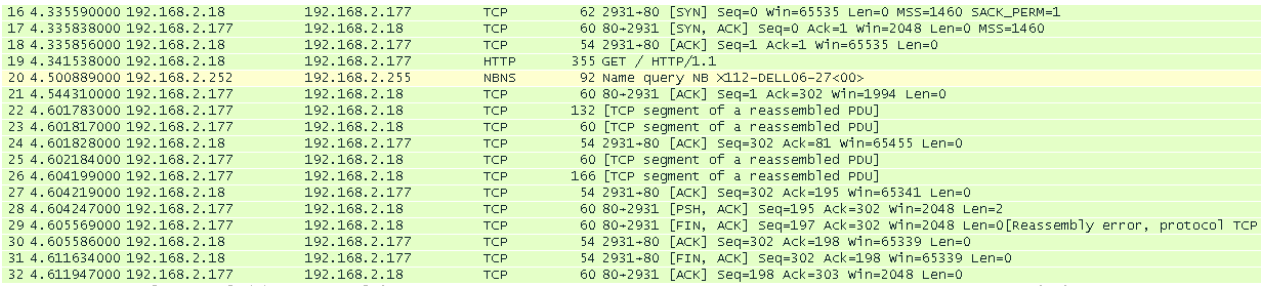
```
26 4.604199000 192.168.2.177   192.168.2.18    TCP    166 [TCP segment of a reassembled PDU]

02 12 00 50 0e 6b b8 7a  4a 07 84 37 80 d8 50 18   ..P.k.z J..7..P.
08 00 4a 1e 00 00 3c 21  44 4f 43 54 59 50 45 20   ..J...<! DOCTYPE
48 54 4d 4c 3e 3c 68 74  6d 6c 20 6c 61 6e 3d 66   HTML><ht ml lan=f
72 3e 3c 68 65 61 64 3e  3c 74 69 74 6c 65 3e 50   r><head> <title>P
61 67 65 20 74 65 6d 6f  69 6e 3c 2f 74 69 74 6c   age temo in</titl
65 3e 3c 2f 68 65 61 64  3e 3c 62 6f 64 79 3e 3c   e></head ><body><
68 31 3e 43 65 6c 61 20  46 6f 6e 63 74 69 6f 6e   h1>Cela  Fonction
6e 65 21 3c 2f 68 31 3e  3c 2f 62 6f 64 79 3e 3c   ne!</h1> </body><
2f 68 74 6d 6c 3e                                  /html>
```

Ci-dessous la fin de la connexion TCP

```
28 4.604247000 192.168.2.177   192.168.2.18    TCP     60 80→2931 [PSH, ACK] Seq=195 Ack=302 Win=2048 Len=2
29 4.605569000 192.168.2.177   192.168.2.18    TCP     60 80→2931 [FIN, ACK] Seq=197 Ack=302 Win=2048 Len=0[Reassembly error, protocol TCP
30 4.605586000 192.168.2.18    192.168.2.177   TCP     54 2931→80 [ACK] Seq=302 Ack=198 Win=65339 Len=0
31 4.611634000 192.168.2.18    192.168.2.177   TCP     54 2931→80 [FIN, ACK] Seq=302 Ack=198 Win=65339 Len=0
32 4.611947000 192.168.2.177   192.168.2.18    TCP     60 80→2931 [ACK] Seq=198 Ack=303 Win=2048 Len=0
```

[FIN-ACK] Le client demande la fin de la connexion
[ACK] le serveur acquitte la fin de la connexion

Synthèse : repérer dans le tableau ci-dessous, les grandes phases de la transmission d'une page web entre un serveur Web et un navigateur Web



16	4.335590000	192.168.2.18	192.168.2.177	TCP	62 2931→80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
17	4.335838000	192.168.2.177	192.168.2.18	TCP	60 80→2931 [SYN, ACK] Seq=0 Ack=1 Win=2048 Len=0 MSS=1460
18	4.335856000	192.168.2.18	192.168.2.177	TCP	54 2931→80 [ACK] Seq=1 Ack=1 Win=65535 Len=0
19	4.341538000	192.168.2.18	192.168.2.177	HTTP	355 GET / HTTP/1.1
20	4.500889000	192.168.2.252	192.168.2.255	NBNS	92 Name query NB X112-DELL06-27<00>
21	4.544310000	192.168.2.177	192.168.2.18	TCP	60 80→2931 [ACK] Seq=1 Ack=302 Win=1994 Len=0
22	4.601783000	192.168.2.177	192.168.2.18	TCP	132 [TCP segment of a reassembled PDU]
23	4.601817000	192.168.2.177	192.168.2.18	TCP	60 [TCP segment of a reassembled PDU]
24	4.601828000	192.168.2.18	192.168.2.177	TCP	54 2931→80 [ACK] Seq=302 Ack=81 Win=65455 Len=0
25	4.602184000	192.168.2.177	192.168.2.18	TCP	60 [TCP segment of a reassembled PDU]
26	4.604199000	192.168.2.177	192.168.2.18	TCP	166 [TCP segment of a reassembled PDU]
27	4.604219000	192.168.2.18	192.168.2.177	TCP	54 2931→80 [ACK] Seq=302 Ack=195 Win=65341 Len=0
28	4.604247000	192.168.2.177	192.168.2.18	TCP	60 80→2931 [PSH, ACK] Seq=195 Ack=302 Win=2048 Len=2
29	4.605569000	192.168.2.177	192.168.2.18	TCP	60 80→2931 [FIN, ACK] Seq=197 Ack=302 Win=2048 Len=0[Reassembly error, protocol TCP
30	4.605586000	192.168.2.18	192.168.2.177	TCP	54 2931→80 [ACK] Seq=302 Ack=198 Win=65339 Len=0
31	4.611634000	192.168.2.18	192.168.2.177	TCP	54 2931→80 [FIN, ACK] Seq=302 Ack=198 Win=65339 Len=0
32	4.611947000	192.168.2.177	192.168.2.18	TCP	60 80→2931 [ACK] Seq=198 Ack=303 Win=2048 Len=0

N°de paquet	Commentaire
16	[SYNC] demande de connexion client->serveur
17	[SYNC-ACK] Acceptation de la demande de connexion serveur -> client
18	[ACK] Acquittement Client->Serveur (le client confirme la connexion TCP, les requêtes http vont pouvoir commencer/
19	Protocole http le client demande une page web avec un GET<chemin> au serveur. Le serveur répond en envoyant des Paquets TCP contenant, entre autre, le code HTML
31	[FIN-ACK] Le client demande la fin de la connexion
32	[ACK] le serveur acquitte la fin de la connexion