M1 Informatique

Protocoles réseaux

TD n° 7

Traces de paquets

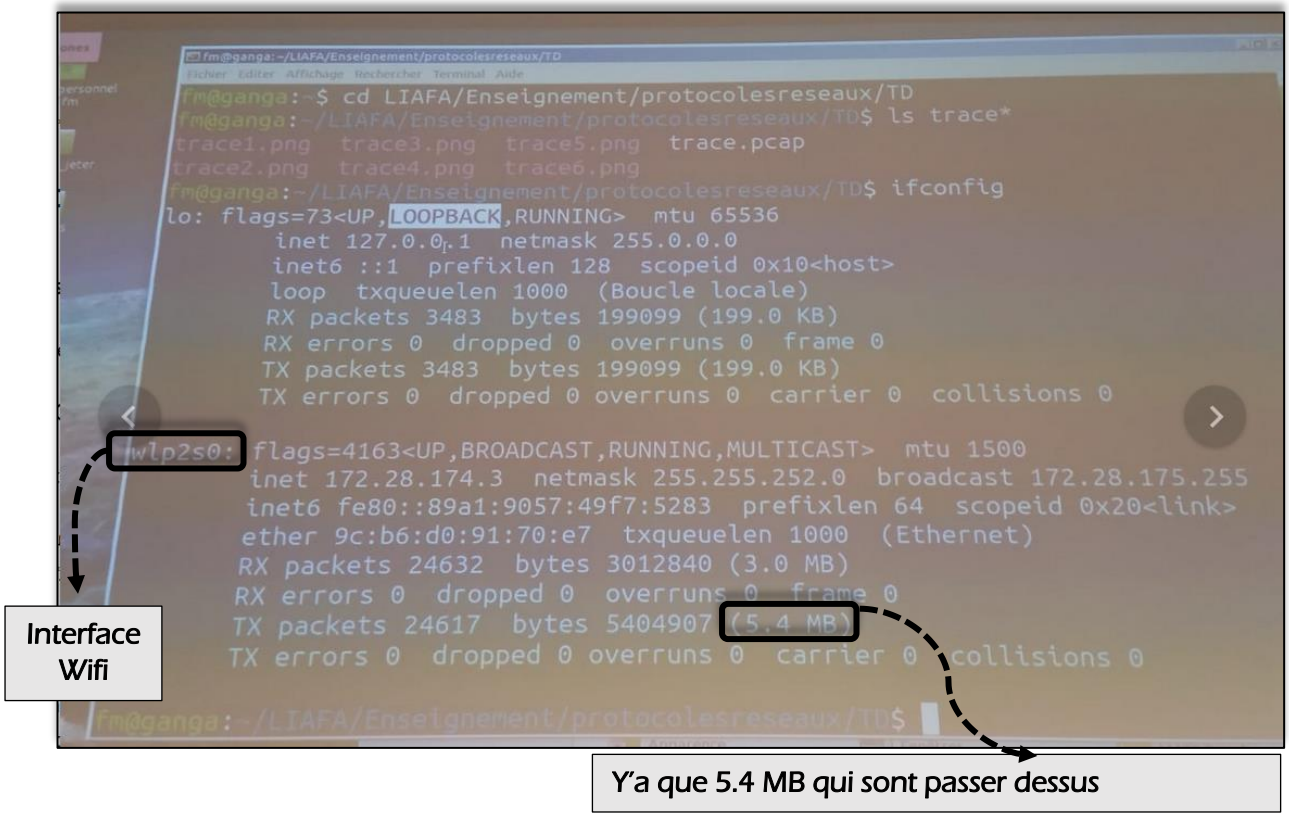
Exercice 1

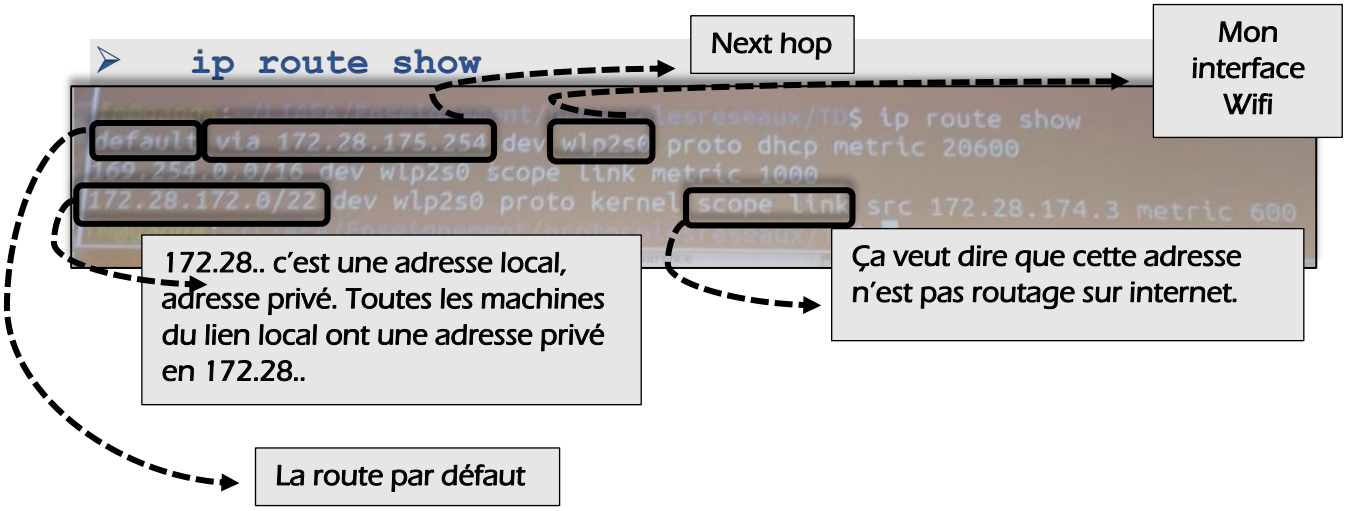
1#

Identifiez le nom de l'interface réseau par laquelle vous êtes connectés à l'Internet. (Sous Linux, vous pouvez par exemple utiliser la commande « **`ip route show`** » pour déterminer l'interface par laquelle passe la route par défaut.)

➢ **`ifconfig`**

On voit les interfaces actives :





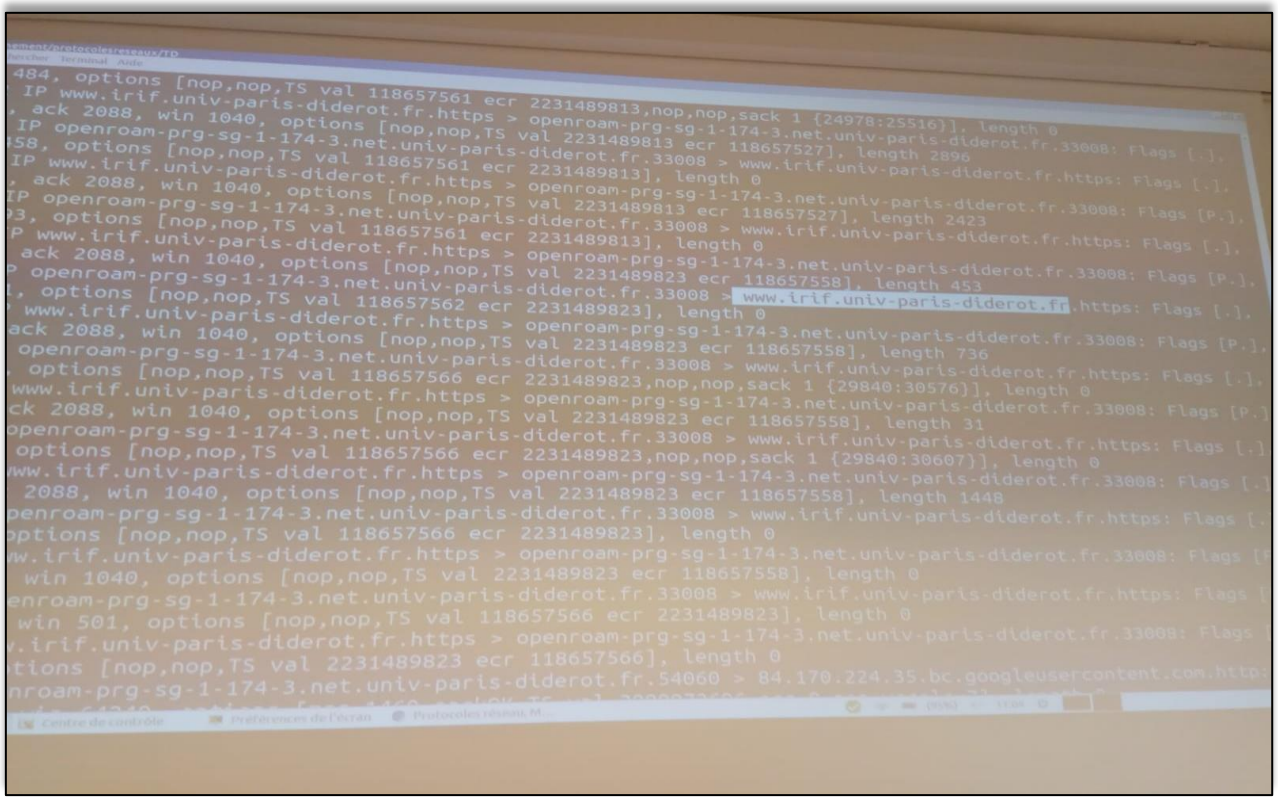
2#

Lancez la commande « `tcpdump -n -i interface` », où `interface` est l'interface déterminée ci-dessus. Pendant que tcpdump s'exécute, chargez une page web. Que se passe-t-il ?

```
➢  tcpdump -n -i wlp2s0
```

- Cette commande va essayer de capturer les paquets qui sorte, cela est possible que pour l'administrateur et il faut donc ajouter `sudo` avant la commande.
- On va retirer `-n` de la commande. Maintenant, je peux voir les machines sur lequelle
- CTRL + C : pour arrêter « tcpdump ».

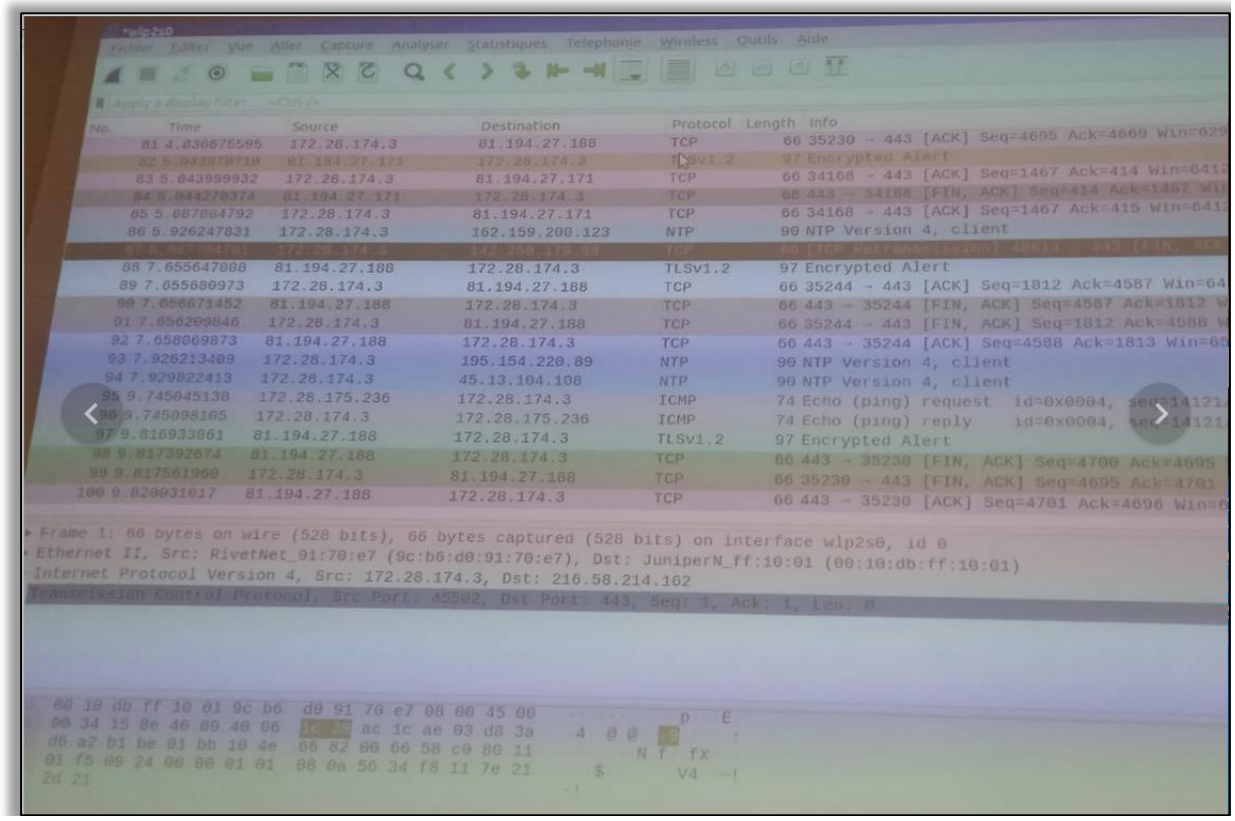
Par exemple, si on navigue vers la page Web du cours on va voir :



3#

Lancez Wireshark. Dans la page d'accueil de Wireshark, choisissez l'interface déterminée ci-dessus. Lancez la capture, puis téléchargez une page web. Que se passe-t-il ?

➢ `wireshark`



Commandes exercice 2

➢ `tcpdump -n -r trace.pcap`

➢ `wireshark trace.pcap`

Exercice 2

Le Téléchargez le fichier

https://www.irif.fr/~jch/enseignement/reseaux/trace.pcap

et examinez-le, d'abord à l'aide de la commande « tcpdump -n -r fichier », ensuite à l'aide de « tcpdump -r fichier », enfin à l'aide de wireshark.

trace.pcap

```
reading from file ../trace.pcap, link-type EN10MB (Ethernet)

10:59:47.105310 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [S], seq 1793060694, win 14600, options [mss
1460,sackOK,TS val 581341 ecr 0,nop,wscale 7], length 0

10:59:47.113894 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [S.], seq 3675933190, ack 1793060695, win 5792,
options [mss 1460,sackOK,TS val 2699823136 ecr 581341,nop,wscale 6], length 0

10:59:47.113990 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 1, win 115, options [nop,nop,TS val 581343
ecr 2699823136], length 0

10:59:47.114230 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [P.], seq 1:118, ack 1, win 115, options [nop,nop,TS
val 581343 ecr 2699823136], length 117: HTTP: GET / HTTP/1.1

10:59:47.125640 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], ack 118, win 91, options [nop,nop,TS val
2699823138 ecr 581343], length 0

10:59:47.125717 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 1:1449, ack 118, win 91, options [nop,nop,TS
val 2699823138 ecr 581343], length 1448: HTTP: HTTP/1.1 200 OK

10:59:47.125753 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 1449, win 137, options [nop,nop,TS val 581346
ecr 2699823138], length 0

10:59:47.128088 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 1449:2897, ack 118, win 91, options
[nop,nop,TS val 2699823138 ecr 581343], length 1448: HTTP

10:59:47.128135 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 2897, win 160, options [nop,nop,TS val 581346
ecr 2699823138], length 0

10:59:47.137330 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 2897:4345, ack 118, win 91, options
[nop,nop,TS val 2699823141 ecr 581346], length 1448: HTTP

10:59:47.137394 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 4345, win 182, options [nop,nop,TS val 581349
ecr 2699823141], length 0

10:59:47.137444 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 4345:5793, ack 118, win 91, options
[nop,nop,TS val 2699823141 ecr 581346], length 1448: HTTP

10:59:47.137459 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 5793, win 205, options [nop,nop,TS val 581349
ecr 2699823141], length 0

10:59:47.139685 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 5793:7241, ack 118, win 91, options
[nop,nop,TS val 2699823141 ecr 581346], length 1448: HTTP

10:59:47.139748 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 7241, win 228, options [nop,nop,TS val 581349
ecr 2699823141], length 0

10:59:47.139812 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 7241:8689, ack 118, win 91, options
[nop,nop,TS val 2699823141 ecr 581346], length 1448: HTTP

10:59:47.139827 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 8689, win 250, options [nop,nop,TS val 581349
ecr 2699823141], length 0

10:59:47.149209 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 8689:10137, ack 118, win 91, options
[nop,nop,TS val 2699823144 ecr 581349], length 1448: HTTP

10:59:47.149295 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 10137, win 273, options [nop,nop,TS val
581352 ecr 2699823144], length 0

10:59:47.149355 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 10137:11585, ack 118, win 91, options
[nop,nop,TS val 2699823144 ecr 581349], length 1448: HTTP

10:59:47.149371 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 11585, win 296, options [nop,nop,TS val
581352 ecr 2699823144], length 0
```

```
10:59:47.149400 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 11585:13033, ack 118, win 91, options [nop,nop,TS val
2699823144 ecr 581349], length 1448: HTTP

10:59:47.149437 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 13033, win 318, options [nop,nop,TS val 581352 ecr
2699823144], length 0

10:59:47.151742 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 13033:14481, ack 118, win 91, options [nop,nop,TS val
2699823144 ecr 581349], length 1448: HTTP

10:59:47.151804 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 14481, win 331, options [nop,nop,TS val 581352 ecr
2699823144], length 0

10:59:47.151862 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 14481:15929, ack 118, win 91, options [nop,nop,TS val
2699823144 ecr 581349], length 1448: HTTP

10:59:47.151878 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 15929, win 320, options [nop,nop,TS val 581352 ecr
2699823144], length 0

10:59:47.154232 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 15929:17377, ack 118, win 91, options [nop,nop,TS val
2699823144 ecr 581349], length 1448: HTTP

10:59:47.154293 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 17377, win 331, options [nop,nop,TS val 581353 ecr
2699823144], length 0

10:59:47.160807 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 17377:18825, ack 118, win 91, options [nop,nop,TS val
2699823147 ecr 581352], length 1448: HTTP

10:59:47.160864 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 18825, win 331, options [nop,nop,TS val 581354 ecr
2699823147], length 0

10:59:47.160918 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 18825:20273, ack 118, win 91, options [nop,nop,TS val
2699823147 ecr 581352], length 1448: HTTP

10:59:47.160932 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 20273, win 320, options [nop,nop,TS val 581354 ecr
2699823147], length 0

10:59:47.160960 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 20273:21721, ack 118, win 91, options [nop,nop,TS val
2699823147 ecr 581352], length 1448: HTTP

10:59:47.163259 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 21721:23169, ack 118, win 91, options [nop,nop,TS val
2699823147 ecr 581352], length 1448: HTTP

10:59:47.163317 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 23169, win 331, options [nop,nop,TS val 581355 ecr
2699823147], length 0

10:59:47.163362 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 23169:24617, ack 118, win 91, options [nop,nop,TS val
2699823147 ecr 581352], length 1448: HTTP

10:59:47.163385 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 24617:26065, ack 118, win 91, options [nop,nop,TS val
2699823147 ecr 581352], length 1448: HTTP

10:59:47.163438 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 26065, win 331, options [nop,nop,TS val 581355 ecr
2699823147], length 0

10:59:47.165913 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 26065:27513, ack 118, win 91, options [nop,nop,TS val
2699823147 ecr 581352], length 1448: HTTP

10:59:47.165960 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 27513:28961, ack 118, win 91, options [nop,nop,TS val
2699823147 ecr 581352], length 1448: HTTP

10:59:47.166035 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 28961, win 331, options [nop,nop,TS val 581356 ecr
2699823147], length 0

10:59:47.166207 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 28961:30409, ack 118, win 91, options [nop,nop,TS val
2699823148 ecr 581353], length 1448: HTTP

10:59:47.168549 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 30409:31857, ack 118, win 91, options [nop,nop,TS val
2699823148 ecr 581353], length 1448: HTTP

10:59:47.168585 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 31857, win 331, options [nop,nop,TS val 581356 ecr
2699823148], length 0

10:59:47.170831 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 31857:33305, ack 118, win 91, options [nop,nop,TS val
2699823150 ecr 581354], length 1448: HTTP

10:59:47.173921 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 33305:34753, ack 118, win 91, options [nop,nop,TS val
2699823150 ecr 581354], length 1448: HTTP

10:59:47.173987 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 34753, win 331, options [nop,nop,TS val 581358 ecr
2699823150], length 0

10:59:47.174039 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 34753:36201, ack 118, win 91, options [nop,nop,TS val
2699823150 ecr 581354], length 1448: HTTP

10:59:47.174067 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 36201:37649, ack 118, win 91, options [nop,nop,TS val
2699823150 ecr 581354], length 1448: HTTP

10:59:47.174132 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 37649, win 331, options [nop,nop,TS val 581358 ecr
2699823150], length 0

10:59:47.176431 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 37649:39097, ack 118, win 91, options [nop,nop,TS val
2699823150 ecr 581354], length 1448: HTTP

10:59:47.176506 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 39097:40545, ack 118, win 91, options [nop,nop,TS val
2699823150 ecr 581354], length 1448: HTTP

10:59:47.176599 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 40545, win 331, options [nop,nop,TS val 581358 ecr
2699823150], length 0
```

```
10:59:47.178704 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 40545:41993, ack 118, win 91, options [nop,nop,TS val
2699823150 ecr 581355], length 1448: HTTP

10:59:47.178751 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [P.], seq 41993:42339, ack 118, win 91, options [nop,nop,TS val
2699823150 ecr 581355], length 346: HTTP

10:59:47.178831 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 42339, win 331, options [nop,nop,TS val 581359 ecr
2699823150], length 0

10:59:47.179679 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [F.], seq 118, ack 42339, win 331, options [nop,nop,TS val 581359 ecr
2699823150], length 0

10:59:47.190218 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [F.], seq 42339, ack 119, win 91, options [nop,nop,TS val 2699823154
ecr 581359], length 0

10:59:47.190284 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags [.], ack 42340, win 331, options [nop,nop,TS val 581362 ecr
2699823154], length 0
```

1#

Quelles sont les adresses des interfaces qui communiquent ?

On voit divers échanges entre **192.168.3.195.39965** (*Numero de port*)

et **138.231.176.10.80** (*Numero de port*) .

```
10:59:47.105310 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags
[S], seq 1793060694, win 14600, options [mss 1460,sackOK,TS val
581341 ecr 0,nop,wscale 7], length 0

10:59:47.113894 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags
[S.], seq 3675933190, ack 1793060695, win 5792, options [mss
1460,sackOK,TS val 2699823136 ecr 581341,nop,wscale 6], length 0
```

2#

Quel protocole de couche transport est-il utilisé ?

Le protocole de la couche transport : TCP

car on voit les numéros de séquence et les numéros d'acquittement. De plus, on voit qu'il y a des options.

TCP *(Transmission Control Protocol)*.

L'en-tête d'un segment TCP comporte également (entre autres) :

- Un **champ de numéro de séquence** et un **champ d'accusé de réception** (tous les deux de 32 bits), utiliser par l'expéditeur et le destinataire TCP dans la mise en œuvre du service de transfert fiable.
- Le **champ d'options** de longueur variable intervient au moment de la négociation de la taille du MSS entre l'expéditeur et le destinataire, ou en tant que facteur d'ajustement de la fenêtre au sein des réseaux a hauts débits. Il existe également une option d'horodatage.

Source : James W. Kurose, Keith W. Ross. Computer Networking, A Top-Down Approach. (Edition française)

```
10:59:47.137330 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags
[.], seq 2897:4345, ack 118, win 91, options [nop,nop,TS val
2699823141 ecr 581346], length 1448: HTTP

10:59:47.137394 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags
[.], ack 4345, win 182, options [nop,nop,TS val 581349 ecr
2699823141], length 0
```

3#

Pourquoi autant de paquets ont-ils une taille égale à 1448 ?

De manière général : limite de 1 500 octets.

Ici, il est écrit au début :

```
reading from file ../trace.pcap, link-type EN10MB (Ethernet)

10:59:47.105310 IP 192.168.3.195.39965 > 138.231.176.10.80:
Flags [S], seq 1793060694, win 14600, options [mss
1460,sackOK,TS val 581341 ecr 0,nop,wscale 7], length 0

10:59:47.113894 IP 138.231.176.10.80 > 192.168.3.195.39965:
Flags [S.], seq 3675933190, ack 1793060695, win 5792, options
[mss 1460,sackOK,TS val 2699823136 ecr 581341,nop,wscale 6],
length 0
```

mss 1460 : données de couche application.

Donc, pourquoi limiter à 1448 ?

Une trame Ethernet : 1, 500 octets

IP	TCP	Charge utile
20 octets	20 octets	1460 octets

Mais l'ajout d'options réduit la taille de 1460 octets :

IP	TCP	Options	Charge utile
20 octets	20 octets	——————1460 octets——————	

Dans les traces de paquets,

- Ce ne sont pas des paquets agrèges, mais de vrais paquets. La limite de capacité étant connu à l'avance, l'émetteur envoie des paquets de taille adapté.
- A la fin : paquet de 346 octets (ce qu'il restai a envoyé après l'envoie de tous les paquets de 1448 octets)

Au total,

Un peu avant la fin du fichier :

```
10:59:47.178751 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags
[P.], seq 41993:42339, ack 118, win 91, options [nop,nop,TS val
2699823150 ecr 581355], length 346: HTTP
```

- Il y a donc 42339-42340 octets qui ont été envoyé
  (Selon le numéro de séquence).
- Petite différence avec la taille du fichier : 42 000 octets [?]

Protocole HTTP (HyperText Transfer Protocol)

Le protocole de transfert utilisé pour le Word Wide Web est HTTP. Il définit les messages que les clients peuvent envoyer au serveur, et ceux que le serveur peut transmettre en réponse. Tous les clients et tous les serveurs doivent respecter les spécifications de ce protocole.

Source : Andrew S. Tanenbaum. Computer Networks.. (Edition française)

4#

Y a-t-il eu des pertes de paquets ?

Un paquet **ACK 40545** ça veut dire quoi ?

```
10:59:47.176599 IP 192.168.3.195.39965 > 138.231.176.10.80:
Flags [.], ack 40545, win 331, options [nop,nop,TS val
581358 ecr 2699823150], length 0
```

Ça veut dire que j'ai reçu les 40545 premières octets, depuis 0 jusqu'à 40544. Prochain octet attendu : 40545.

De manière général,

Si on reçoit le segment **[0 – 1000 [** et après on reçois le segment **[2000 – 3000 [**, alors on sait que **[1000 – 2000 [** est perdu.

[0 – 1000 [*recu* ⟶ *ack* **1000**

[1000 – 2000 [*perdu*

[2000 – 3000 [*recu* ⟶ *ack* **1000**

[1000 – 2000 [*recu* ⟶ *ack* **3000**

ack dupliqué signale une perte

S'il n'y aurai pas de pertes : le ACK suivant aurai été un ACK de 2000.

Donc, y a-t-il eu des pertes de paquets ?

A compléter sur le pdf.

5#

Pourquoi autant de ack 118 ?

Pourquoi autant de ack 118 ? Est-ce que ça signale qu'il y a autant de pertes de paquets ?

- ACK dupliqué = perte ; C'est vrai.
- Mais il y a un autre mécanisme qui peut expliquer les ACK dupliquée : on voit qu'il y a beaucoup de paquets de longueur nulle (0) ce qui n'incrémente pas le nombre d'acquittements.
- A quoi ça sert d'envoyer un paquet de longueur 0 ? le paquet contient des options mais surtout il transmet un numéro de récepteur et autres données.
    - tcpdump n'affiche pas les numéros de séquence de paquets de longueur 0.

```
10:59:47.149400 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 11585:13033, ack 118, win 91, options [nop,nop,TS val 2699823144 ecr 581349], length 1448: HTTP
```

win = taille de la fenêtre du récepteur, ca règle la vitesse.

Si la valeur est 0 : STOP, demande d'arrêt d'envoi de donnes.

```
10:59:47.160960 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 20273:21721, ack 118, win 91, options [nop,nop,TS val 2699823147 ecr 581352], length 1448: HTTP
10:59:47.163259 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 21721:23169, ack 118, win 91, options [nop,nop,TS val 2699823147 ecr 581352], length 1448: HTTP
```

2 chiffres pour avoir plus de lisibilité.

6#

Pourquoi les numéros de séquence des deux premiers paquets sont énormes, et ceux des suivants petits ?

Par exemple, on a :

```
10:59:47.163259 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags [.], seq 21721:23169, ack 118, win 91, options [nop,nop,TS val 2699823147 ecr 581352], length 1448: HTTP
```

Mais, pourquoi au début les numéros de séquence des deux premiers paquets sont énormes ?

```
10:59:47.105310 IP 192.168.3.195.39965 > 138.231.176.10.80: Flags
[S], seq 1793060694, win 14600, options [mss 1460,sackOK,TS val
581341 ecr 0,nop,wscale 7], length 0

10:59:47.113894 IP 138.231.176.10.80 > 192.168.3.195.39965: Flags
[S.], seq 3675933190, ack 1793060695, win 5792, options [mss
1460,sackOK,TS val 2699823136 ecr 581341,nop,wscale 6], length 0
```

En réalité les numéros de séquence de paquets sont renommés par **tcpdump**, pour la lisibilité.

Numéros de séquence

Le champ d'en-tête de numéro de séquence et d'accusé de réception des segments TCP jouent un rôle fondamental dans le service de transfert de données fiable de TCP.

Au regard de TCP, les données à transférer se présentent sous forme de flux sans structure, mais utilisant un ordre précis. Ces numéros de séquence TCP s'appliquent aux différents flux d'octets et **non** aux différents segments génères pour leur transfert.

Le **numéro de séquence d'un segment** est ainsi le numéro dans le flux d'octets du premier octet de chaque segment. Soit un processus exploité sur le serveur A qui souhaite envoyer un flux de donnes à un processus du serveur B sur la connexion TCP. Le TCP du serveur A procède automatiquement à la numérotation de tous les octets du flux de données. Supposons que le flux de données consiste en un fichier de 500 Ko, que la taille du MMS soit fixée à 1 Ko et que le premier octet du flux de données ait le numéro séquence 0. TCP scindera les flux de données en 500 segments (voir figure 3.30). Le premier segment aura le numéro 0, le deuxième le numéro 1000, le troisième le numéro 2000, etc. Chaque numéro de séquence est inséré dans le champ de numéro de séquence de l'en-tête du segment TCP approprié.

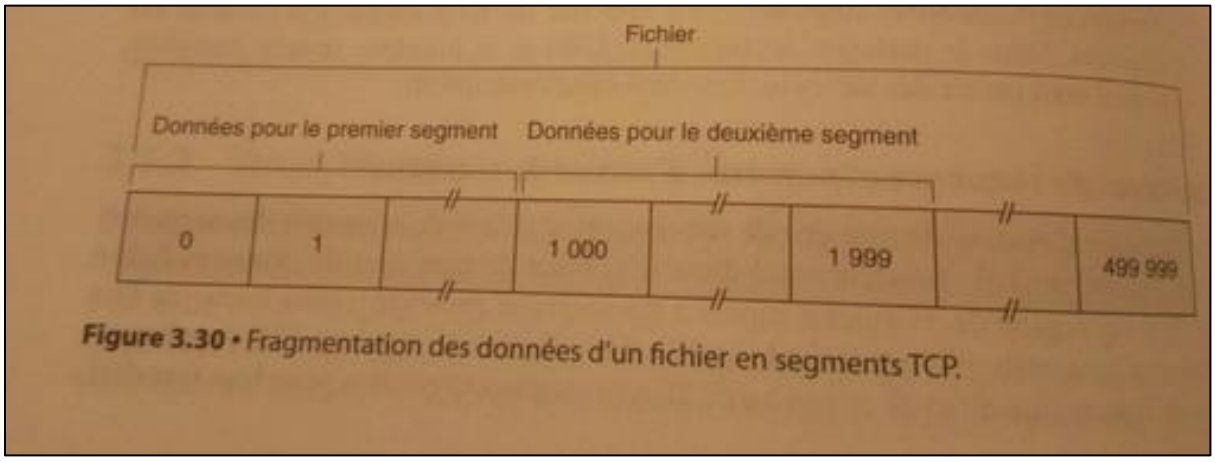


Figure 3.30 • Fragmentation des données d'un fichier en segments TCP.

Source : James W. Kurose, Keith W. Ross. Computer Networking, A Top-Down Approach. (Edition française)

7#

Au fait, quel est le nom des hôtes qui communiquent ?

8#

Interpréter les flags (lettre entre crochets comme [S.])

Les flags c'est un truc de TCP.

- [.]
- [p.] – **Push :** au niveau application on a fini d'envoyer les données.
- [F.] – **Finalise :** Fin de la communication.
- [s.]
- [S.]

- Lors de la connexion : avec le flag s.
- F = Finalise, fin de la transmission, paquets pour raccrocher.
  En plus, chaque un des paquets de fin doit être acquittée.
- Et en même temp on ferme la communication dans le sens serveur-client.

9#

Vers la fin on dirait qu'il y a moins de ***ack***. Pourquoi ?

- Alors qu'on regarde l'échange serveur vers client : au début on a un ping-pong paquet-acquittement, et a la fin on a paquets-acquittements.
- En faite TCP n'oblige pas d'acquitter chaque paquet, donc si débit et latence élevé il y a beaucoup de paquets alors si on envoie un acquittement a chaque fois c'est pas la pagine. On a la mécanisme de l' acquittement retardé (*delayed ack*).

10#

Interprétez les champs val et ecr.

val, ecr : Données d'horodatage.
On mais un numéro aléatoire ..... comme ca on peut calculer la latence de la ligne...
Quand on envoie un certain nombre X, et on reçoie, un écho d'un certain nombre X.
Avec le mécanisme de perte et ack retardé, le numéro de séquence est pas suffisant pour calculer la latence de la ligne.

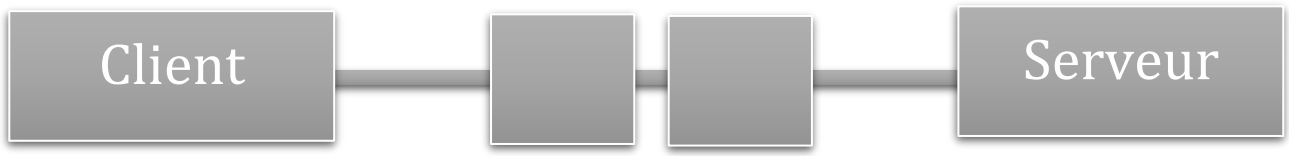
L'option nop - purage. Qu'on c'est pas un multiple de 4 on ajoute nop pour avoir un multiple de 4.

Il semble qu'il manque un champ de type $temp$.

11#

Sur quel hôte la trace a-t-elle été capturée ? Justifier.

La trace peut être capturée sur divers machines :



Des trucs au milieu

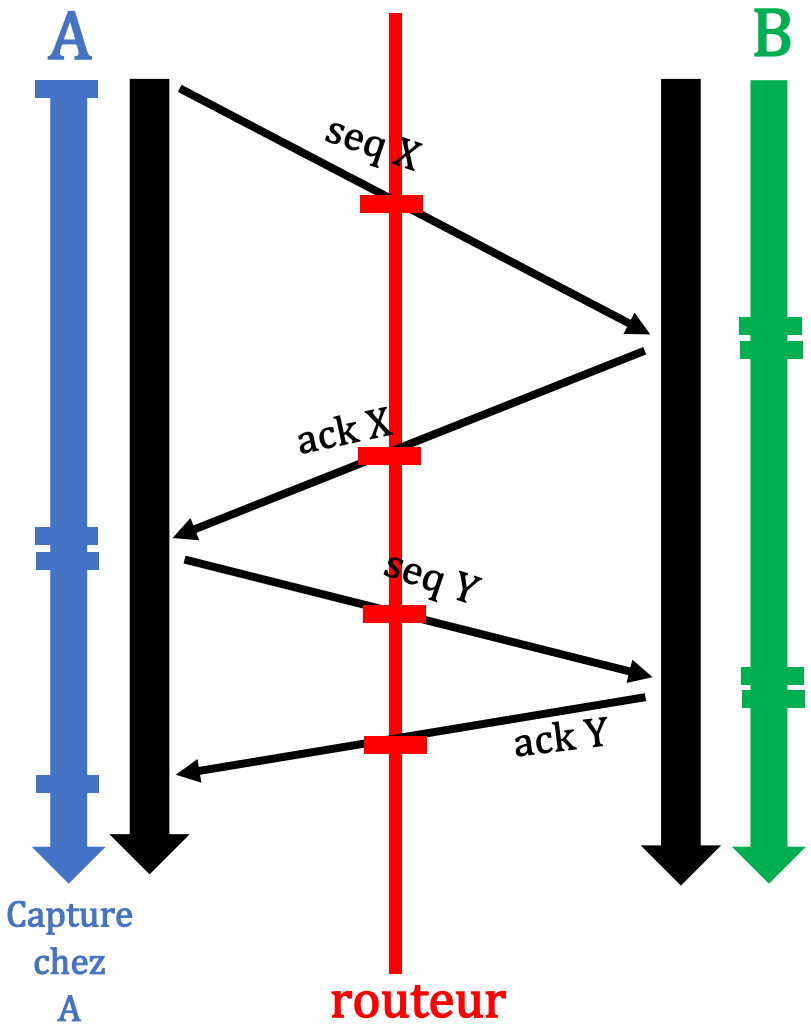
La capture a u lieu sur quelle machine ?

L'évidence est que c'est plus facile de capturer sur le client que sur le serveur ou les routeurs. Donc ca semble logique que ca soit le client. Mais comment prouver ça ?

On voit que le temp entre la réception du paquet et l'émission de l'acquittement correspondant : _____

Si je fait un pronogrames des paquets ?

Si je capture sur la machine a l'eurodatage que j'ai est comme ça :



Donc, capturé chez les clients web car :

(1) Simplicité
(2) Selon l'eurodatage (le vrai argument)

Exercice 3

#

Que se passe-t-il dans les fragments de traces suivants ? Toutes les traces ont été capturées sur la machine « A » (comment peut-on le voir ?).

- ***tcpdump*** affiche pas exactement de la même façon qu'ici. Par exemple, la longueur est donnée entre parenthèses.
- On voit des ***win*** : 5 échanges TCP.

1.

```
22:45:26.121149 A.32769 > B.www: S 2919412148:2919412148(0) win 5840
                                   <mss 1460,nop,wscale 1>

22:45:26.123055 B.www > A.32769: S 4144006771:4144006771(0)
                                   ack 2919412149 win 65535
                                   <mss 1460,nop,wscale 1>
22:45:26.123120 A.32769 > B.www: . ack 1 win 2920
22:45:26.124144 A.32769 > B.www: P 1:146(145) ack 1 win 2920
22:45:26.130323 B.www > A.32769: . 1:1461(1460) ack 146 win 32850
22:45:26.130402 A.32769 > B.www: . ack 1461 win 4380
22:45:26.130750 B.www > A.32769: . 1461:2921(1460) ack 146 win 32850
```

- La séquence 1, elle ressemble à quoi ? A quoi ces 7 paquets nous font penser ?
- Une question a ce poser à chaque fois : on est au début / milieu / fin de la capture ?
- Donc on est au début et la on voit que ça ressemble au traces déjà vu : serveur http qui répond à un client http.

- Le 4eme paquet : on voit que il y a 145 octets, et après 1460...
- Les 2 premiers segment de la réponses qui font 1460.

- B – serveur web.
  A – client web (au début il fait un get).
- http c'est l'exemple tipe de requête-réponse, il y a des protocoles plus bavards..

2.

```
22:45:26.195481 B.www > A.32769: . 74461:75921(1460) ack 146 win 32850
22:45:26.196216 B.www > A.32769: P 75921:77381(1460) ack 146 win 32850
22:45:26.196228 A.32769 > B.www: . ack 77381 win 64240
22:45:26.211281 B.www > A.32769: . 77381:78841(1460) ack 146 win 32850
22:45:26.211772 B.www > A.32769: P 78841:80301(1460) ack 146 win 32850
22:45:26.211783 A.32769 > B.www: . ack 80301 win 64240
```

- Il y a de l'acquittement tardé.

3.

```
22:45:26.240764 B.www > A.32769: P 127021:128387(1366) ack 146 win 32850
22:45:26.240776 A.32769 > B.www: . ack 128387 win 64240
22:45:26.249437 A.32769 > B.www: F 146:146(0) ack 128387 win 64240
22:45:26.251418 B.www > A.32769: . ack 147 win 32850
22:45:26.251840 B.www > A.32769: F 128387:128387(0) ack 147 win 32850
22:45:26.251871 A.32769 > B.www: . ack 128388 win 64240
```

- Il y a les **F** donc c'est la fin du transfert.
- A ferme la communication ; B acquitte.
- B ferme la communication ; A acquitte.

- Une socket TCP c'est une socket bi-directionnelle.

4.

```
22:49:22.739301 A.32775 > C.ssh: . 110064:111524(1460) ack 2336 win 5104
22:49:22.739312 A.32775 > C.ssh: . 111524:112984(1460) ack 2336 win 5104
22:49:22.772816 C.ssh > A.32775: . ack 96924 win 62780
22:49:22.772828 A.32775 > C.ssh: . 112984:114444(1460) ack 2336 win 5104
22:49:22.772838 A.32775 > C.ssh: . 114444:115904(1460) ack 2336 win 5104
22:49:22.773905 C.ssh > A.32775: . ack 99844 win 62780
```

- SSH c'est aussi pour ce connecté a une machine a distance, mais ajd ça peut avoir des significations plus larges, on peut transférer bcp de choses via SSH, c'est une sorte de sous-couche.

5.

```
22:53:47.759896 D.17775 > A.32782: . 75921:77381(1460) ack 1 win 17520
22:53:47.760031 D.17775 > A.32782: . 77381:78841(1460) ack 1 win 17520
22:53:47.760055 A.32782 > D.17775: . ack 78841 win 64240
22:53:47.885001 D.17775 > A.32782: . 80301:81761(1460) ack 1 win 17520
22:53:47.885072 A.32782 > D.17775: . ack 78841 win 64240
22:53:47.885816 D.17775 > A.32782: . 83221:84681(1460) ack 1 win 17520
22:53:47.885834 A.32782 > D.17775: . ack 78841 win 64240
22:53:47.885951 D.17775 > A.32782: . 84681:86141(1460) ack 1 win 17520
22:53:47.885962 A.32782 > D.17775: . ack 78841 win 64240
22:53:47.917054 D.17775 > A.32782: . 86141:87601(1460) ack 1 win 17520
22:53:47.917065 A.32782 > D.17775: . ack 78841 win 64240
22:53:48.042369 D.17775 > A.32782: . 78841:80301(1460) ack 1 win 17520
22:53:48.042419 A.32782 > D.17775: . ack 81761 win 64240
22:53:48.199537 D.17775 > A.32782: . 81761:83221(1460) ack 1 win 17520
22:53:48.199602 A.32782 > D.17775: . ack 87601 win 64240
22:53:48.199718 D.17775 > A.32782: . 89061:90521(1460) ack 1 win 17520
22:53:48.199732 A.32782 > D.17775: . ack 87601 win 64240
22:53:48.356490 D.17775 > A.32782: . 87601:89061(1460) ack 1 win 17520
22:53:48.356551 A.32782 > D.17775: . ack 90521 win 64240
22:53:48.356620 D.17775 > A.32782: . 90521:91981(1460) ack 1 win 17520
```

- Transfert uni-directionnelle.
- Il y a des pertes de paquets.
- L'ordre a lequelle les segment sont envoyée est pas croissant car faut rattraper les pertes.
- Cette échange ce fait sur une échange pas très stable.