

## Rapport de TP : Réseaux

### Analyse de trames

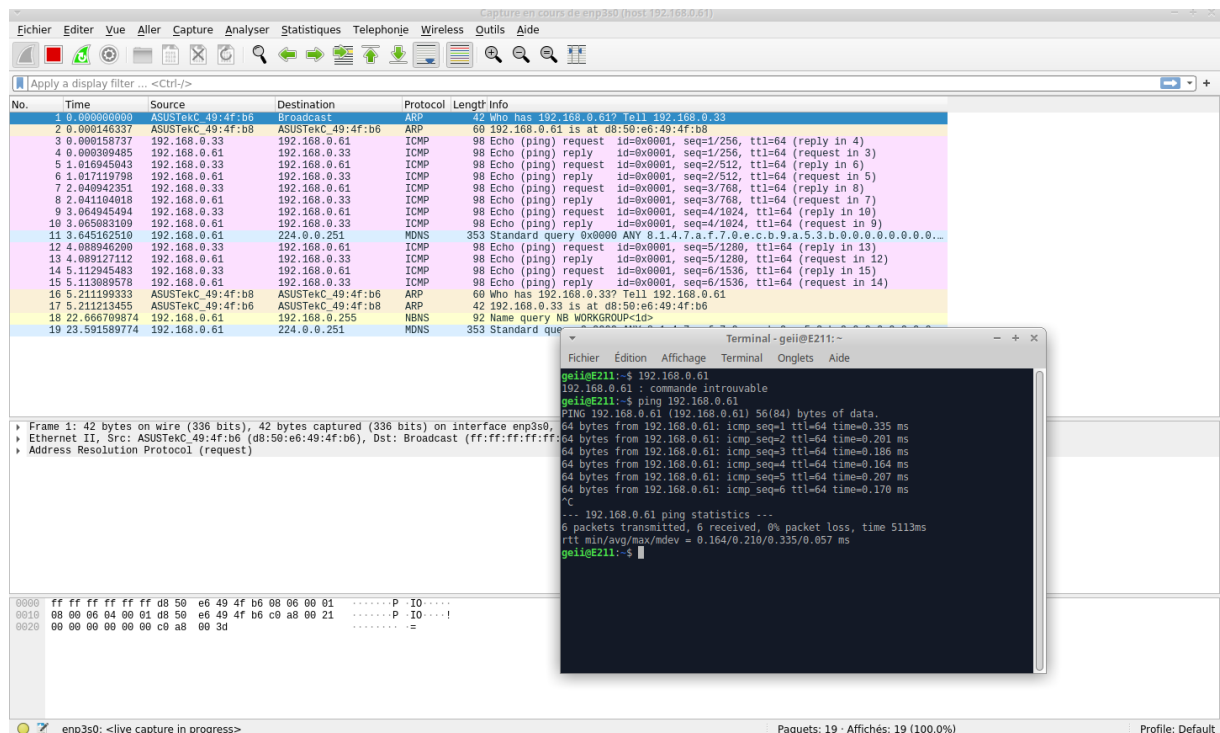
Au cours de ce TP, nous avons utilisé le logiciel Wireshark pour pouvoir capturer et analyser les trames Ethernets.

#### Captures ARP et ICMP :

Quand nous lançons l'application pour la 1<sup>ère</sup> fois, nous pouvons observer une très grande quantité de requetes ARP sur l'adresse de diffusion (192.168.0.255)

Capture en cours de enp350						
Fichier	Editor	Vue	Aller	Capture	Analyses	Statistiques
Telephonie	Wireless	Qutils	Aide			

Après avoir vider la table des entrées arp (*sudo arp -d adresse*), nous pouvons effectuer un ping sur une machine et observer le trafic qui en résulte :



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	ASUSTeK 40:4f:b6	Broadcast	ARP	42	Who has 192.168.0.61? Tell 192.168.0.33
2	0.000146337	ASUSTeK 40:4f:b6	ASUSTeK 40:4f:b6	ARP	60	192.168.0.61 is at 08:50:e6:49:4f:b6
3	0.000158737	192.168.0.33	192.168.0.61	ICMP	98	Echo (ping) request id=0x0001, seq=1/256, ttl=64 (request in 4)
4	0.000309485	192.168.0.61	192.168.0.33	ICMP	98	Echo (ping) reply id=0x0001, seq=1/256, ttl=64 (request in 3)
5	1.016945043	192.168.0.33	192.168.0.61	ICMP	98	Echo (ping) request id=0x0001, seq=2/512, ttl=64 (request in 6)
6	1.017119798	192.168.0.61	192.168.0.33	ICMP	98	Echo (ping) reply id=0x0001, seq=2/512, ttl=64 (request in 5)
7	2.040942351	192.168.0.33	192.168.0.61	ICMP	98	Echo (ping) request id=0x0001, seq=3/768, ttl=64 (request in 8)
8	2.041104018	192.168.0.61	192.168.0.33	ICMP	98	Echo (ping) reply id=0x0001, seq=3/768, ttl=64 (request in 7)
9	3.064945494	192.168.0.33	192.168.0.61	ICMP	98	Echo (ping) request id=0x0001, seq=4/1024, ttl=64 (request in 9)
10	3.065083100	192.168.0.61	192.168.0.33	ICMP	98	Echo (ping) reply id=0x0001, seq=4/1024, ttl=64 (request in 9)
11	3.645102510	192.168.0.61	224.0.0.251	NDNS	353	Standard query 0x0000 ANY 8.1.4.7.a.f.7.0.e.c.b.9.a.5.3.b.0.0.0.0.0.0.0...
12	4.080946200	192.168.0.33	192.168.0.61	ICMP	98	Echo (ping) request id=0x0001, seq=5/1280, ttl=64 (request in 13)
13	4.080912712	192.168.0.61	192.168.0.33	ICMP	98	Echo (ping) reply id=0x0001, seq=5/1280, ttl=64 (request in 12)
14	5.112945403	192.168.0.33	192.168.0.61	ICMP	98	Echo (ping) request id=0x0001, seq=6/1536, ttl=64 (request in 15)
15	5.113089578	192.168.0.61	192.168.0.33	ICMP	98	Echo (ping) reply id=0x0001, seq=6/1536, ttl=64 (request in 14)
16	5.211199333	ASUSTeK 40:4f:b6	ASUSTeK 40:4f:b6	ARP	60	Who has 192.168.0.33? Tell 192.168.0.61
17	5.212134555	ASUSTeK 40:4f:b6	ASUSTeK 40:4f:b6	ARP	42	192.168.0.33 is at 08:50:e6:49:4f:b6
18	22.606709074	192.168.0.61	192.168.0.255	NDNS	92	Name query NS WORKGROUP-ID>
19	23.591589774	192.168.0.61	224.0.0.251	NDNS	353	Standard query 0x0000 ANY 8.1.4.7.a.f.7.0.e.c.b.9.a.5.3.b.0.0.0.0.0.0.0...

```
geileE211:~  
geileE211:~$ ping 192.168.0.61  
PING 192.168.0.61 (192.168.0.61) 56(84) bytes of data:  
64 bytes from 192.168.0.61: icmp_seq=1 ttl=64 time=0.335 ms  
64 bytes from 192.168.0.61: icmp_seq=2 ttl=64 time=0.201 ms  
64 bytes from 192.168.0.61: icmp_seq=3 ttl=64 time=0.186 ms  
64 bytes from 192.168.0.61: icmp_seq=4 ttl=64 time=0.164 ms  
64 bytes from 192.168.0.61: icmp_seq=5 ttl=64 time=0.207 ms  
64 bytes from 192.168.0.61: icmp_seq=6 ttl=64 time=0.170 ms  
^C  
--- 192.168.0.61 ping statistics ---  
6 packets transmitted, 6 received, 0% packet loss, time 5113ms  
rtt min/avg/max/mdev = 0.164/0.210/0.335/0.057 ms  
geileE211:~$
```

Nous pouvons observer que le ping commence par une diffusion d'une requête arp (Address Resolution Protocol), demandant l'adresse MAC (ou adresse physique) affiliée à l'adresse IP 192.168.0.61. On trouve aussi dans cette trame l'adresse source (notre ordinateur):

Time	Source	Destination	Protocol	Length	Info
1 0.000000000	ASUSTekC_49:4f:b6	Broadcast	ARP	42	Who has 192.168.0.61? Tell 192.168.0.33

On peut d'ailleurs observer que la taille de la trame est de 42 octets, contre les 60 octets minimum demandés (les 4 octets du CRC32 ne sont pas comptés par Wireshark). Pour cela, il faut ajouter du bourrage (ou padding), mais celui est ajouté après la capture de la trame, c'est pour cela que nous ne le voyons pas. Cependant, si on regarde les trames arrivant sur la machine cible, on peut mettre en évidence la présence de padding :

2 5.435896046	ASUSTekC_49:4f:b6	Broadcast	ARP	60	Who has 192.168.0.61? Tell 192.168.0.33
---------------	-------------------	-----------	-----	----	---

Un exemple de padding dans une trame :

Sans padding :

ff ff ff ff ff ff d8 50 e6 49 4f b6 08 06 00 01	.....P IO.....
08 00 06 04 00 01 d8 50 e6 49 4f b6 c0 a8 00 21	.....P IO.....!
00 00 00 00 00 00 c0 a8 00 3d	.....=

Avec padding :

ff ff ff ff ff ff d8 50 e6 49 4f b6 08 06 00 01	.....P IO.....
08 00 06 04 00 01 d8 50 e6 49 4f b6 c0 a8 00 21	.....P IO.....!
00 00 00 00 00 00 c0 a8 00 3d {00 00 00 00 00 00	.....=.....
00 00 00 00 00 00 00 00 00 00 } padding	.....

Ensuite vient la réponse de l'ordinateur cible, qui réponds en confirmant son adresse IP et son adresse MAC (les 2 adresses sont spécifiés pour éviter les attaques de type Man In The Middle) :

2 0.000146337	ASUSTekC_49:4f:b8	ASUSTekC_49:4f:b6	ARP	60	192.168.0.61 is at d8:50:e6:49:4f:b8
---------------	-------------------	-------------------	-----	----	--------------------------------------

On obtient donc l'adresse mac du pc ciblé, qui est bien celle que l'on retrouve avec la commande *ifconfig -a*. Si l'on regarde à nouveau la table des entrées ARP, on remarque qu'une nouvelle ligne est apparu, avec l'adresse de la machine ciblée (ici 192.168.0.1). Un exemple réalisé avec 2 machines sur mon réseau privé :

Avant le ping :

```
[root@parrot05]~/home/tihmz
#arp -n
```

Adresse	TypeMap	AdresseMat	Indicateurs	Iface
192.168.1.44	ether	00:60:6e:33:73:5c	C	wlan0
192.168.1.10	ether	00:e0:4c:80:b7:63	C	wlan0

Après le ping de 192.168.1.118 :

```
[root@parrot05]~/home/tihmz
#arp -n
```

Adresse	TypeMap	AdresseMat	Indicateurs	Iface
192.168.1.44	ether	00:60:6e:33:73:5c	C	wlan0
192.168.1.10	ether	00:e0:4c:80:b7:63	C	wlan0
192.168.1.118	ether	d8:c4:97:c9:ee:b6	C	wlan0

Ainsi, après que notre machine ait récupéré l'adresse mac de l'ordinateur ciblé, nous pouvons commencer le test de connectivité avec le protocole ICMP (pour Internet Control Message Protocol). Grâce à ce protocole, nous pouvons tester la connectivité entre 2 appareils en envoyant des trames d'octets à la machine ciblée. Dans notre cas, on envoi ici des trames de type « Echo Request », qui servent juste à tester l'accessibilité d'un appareil dans un réseau. Il existe différent type de trame pour un ping, telle que Traceroute (suivit du paquet à travers différents routeur), Timestamp (demande d'heure), etc.

3	0.000158737	192.168.0.33	192.168.0.61	ICMP	98 Echo (ping) request	id=0x0001, seq=1/256, ttl=64 (reply in 4)
4	0.000399485	192.168.0.61	192.168.0.33	ICMP	98 Echo (ping) reply	id=0x0001, seq=1/256, ttl=64 (request in 3)
5	1.016945043	192.168.0.33	192.168.0.61	ICMP	98 Echo (ping) request	id=0x0001, seq=2/512, ttl=64 (reply in 6)
6	1.017119798	192.168.0.61	192.168.0.33	ICMP	98 Echo (ping) reply	id=0x0001, seq=2/512, ttl=64 (request in 5)
7	2.040942351	192.168.0.33	192.168.0.61	ICMP	98 Echo (ping) request	id=0x0001, seq=3/768, ttl=64 (reply in 8)

On remarque aussi que lors de l'interruption du ping, on retrouve nos 2 requêtes ARP de vérification.

On peut d'ailleurs filtrer les résultats pour ne voir que le trames ARP ou ICMP par exemple :

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	ASUSTekC_49:4f:b6	Broadcast	ARP	42	Who has 192.168.0.61? Tell 192.168.0.33
2	0.000146337	ASUSTekC_49:4f:b8	ASUSTekC_49:4f:b6	ARP	60	192.168.0.61 is at d8:50:e6:49:4f:b8
16	5.211199333	ASUSTekC_49:4f:b8	ASUSTekC_49:4f:b6	ARP	60	Who has 192.168.0.33? Tell 192.168.0.61
17	5.211213455	ASUSTekC_49:4f:b6	ASUSTekC_49:4f:b8	ARP	42	192.168.0.33 is at d8:50:e6:49:4f:b6
24	124.697731356	ASUSTekC_49:4f:b6	Broadcast	ARP	42	Who has 192.168.0.61? Tell 192.168.0.33
25	124.697859413	ASUSTekC_49:4f:b8	ASUSTekC_49:4f:b6	ARP	60	192.168.0.61 is at d8:50:e6:49:4f:b8
35	129.878893750	ASUSTekC_49:4f:b8	ASUSTekC_49:4f:b6	ARP	60	Who has 192.168.0.33? Tell 192.168.0.61
36	129.878913865	ASUSTekC_49:4f:b6	ASUSTekC_49:4f:b8	ARP	42	192.168.0.33 is at d8:50:e6:49:4f:b6

Si l'on refait ce même test, avec cette fois une trame de 1800octets(*ping adresse -s 1800*), on peut observer la fragmentation des paquets (car au-dessus du Maximum Transfert Unit de 1500octets) :

No.	Time	Source	Destination	Protocol	Length	Info
50	623.852416113	192.168.0.61	224.0.0.251	NDNS	354	Standard query 0x0000 ANY 8.1.4.7.a.f.7.0.e.c.b.9.a.5.3.b.0.0.0.0.0.0.0...
51	627.011122734	192.168.0.61	192.168.0.255	NDNS	92	Name query NB WORKGROUP<id>
52	647.007686152	ASUSTekC_49:4f:b8	Broadcast	ARP	60	Who has 192.168.0.33? Tell 192.168.0.61
53	647.007704997	ASUSTekC_49:4f:b6	ASUSTekC_49:4f:b8	ARP	42	192.168.0.33 is at d8:50:e6:49:4f:b6
54	647.008047259	192.168.0.61	192.168.0.33	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=d53a) [Reassembled in #55]
55	647.008047884	192.168.0.61	192.168.0.33	ICMP	362	Echo (ping) request id=0x0003, seq=1/256, ttl=64 (reply in 57)
56	647.008103464	192.168.0.33	192.168.0.61	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=7db8) [Reassembled in #57]
57	647.008138997	192.168.0.33	192.168.0.61	ICMP	362	Echo (ping) reply id=0x0003, seq=1/256, ttl=64 (request in 55)
58	648.101239991	192.168.0.61	192.168.0.33	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=d5b8) [Reassembled in #59]
59	648.101249533	192.168.0.61	192.168.0.33	ICMP	362	Echo (ping) request id=0x0003, seq=2/512, ttl=64 (reply in 61)
60	648.101279531	192.168.0.33	192.168.0.61	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=7c32) [Reassembled in #61]
61	648.101283403	192.168.0.33	192.168.0.61	ICMP	362	Echo (ping) reply id=0x0003, seq=2/512, ttl=64 (request in 59)
62	649.125207054	192.168.0.61	192.168.0.33	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=d687) [Reassembled in #63]
63	649.125207651	192.168.0.61	192.168.0.33	ICMP	362	Echo (ping) request id=0x0003, seq=3/768, ttl=64 (reply in 65)
64	649.125257114	192.168.0.33	192.168.0.61	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=7c78) [Reassembled in #65]
65	649.125269939	192.168.0.33	192.168.0.61	ICMP	362	Echo (ping) reply id=0x0003, seq=3/768, ttl=64 (request in 63)
66	650.153170998	192.168.0.61	192.168.0.33	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=d72f) [Reassembled in #67]
67	650.153170998	192.168.0.61	192.168.0.33	ICMP	362	Echo (ping) request id=0x0003, seq=4/1024, ttl=64 (reply in 69)
68	650.153218366	192.168.0.33	192.168.0.61	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=7c7c) [Reassembled in #69]
69	650.153223683	192.168.0.33	192.168.0.61	ICMP	362	Echo (ping) reply id=0x0003, seq=4/1024, ttl=64 (request in 67)
70	651.173134013	192.168.0.61	192.168.0.33	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=d817) [Reassembled in #71]
71	651.173134553	192.168.0.61	192.168.0.33	ICMP	362	Echo (ping) request id=0x0003, seq=5/1280, ttl=64 (reply in 73)
72	651.173182219	192.168.0.33	192.168.0.61	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=7d2f) [Reassembled in #73]
73	651.173187223	192.168.0.33	192.168.0.61	ICMP	362	Echo (ping) reply id=0x0003, seq=5/1280, ttl=64 (request in 71)
74	652.088990662	ASUSTekC_49:4f:b6	ASUSTekC_49:4f:b8	ARP	42	Who has 192.168.0.61? Tell 192.168.0.33
75	652.0889925619	ASUSTekC_49:4f:b8	ASUSTekC_49:4f:b6	ARP	60	192.168.0.61 is at d8:50:e6:49:4f:b8

## Le protocole TCP :

Le protocole TCP (Transmission Control Protocol) est un protocole « connection oriented » : cela signifie qu'une connexion doit être établie entre une machine et le serveur avant qu'il puisse y avoir un transfert de données. Le serveur écoute les demandes de connexion sur un port spécifique (on dit que c'est un port ouvert). Dans notre cas, le serveur tourne sur une machine appelée « Chuck » à l'adresse 192.168.0.25 sur le port 10000. Pour pouvoir se connecter, on utilise la commande *telnet*.

Telnet (pour TELEcommunication network) est un protocole permettant de communiquer avec un serveur distant. Très utilisé autrefois, il est aujourd'hui obsolète (peu sécurisé, les données échangées étant non cryptées) et est aujourd'hui remplacé par le protocole SSH (Secure Shell).

Cependant, on peut encore utiliser le client telnet pour établir des connexions TCP avec un serveur distant.

Avec la commande *telnet adresse port*, on peut se connecter au serveur, qui nous renvoie des données (ici, l'heure) :

```
geii@E211:~$ telnet 192.168.0.25 10000
Trying 192.168.0.25...
Connected to 192.168.0.25.
Escape character is '^]'.

Bienvenue sur le serveur de temps: 192.168.0.33 => 11h 26min 50s
```

Sur WireShark, on peut observer les connexions :

The image shows a Wireshark packet capture and a terminal window. The terminal window displays the output of a telnet command: `geii@E211:~$ telnet 192.168.0.25 10000`. The output shows the connection attempt, the connection being established, and the server responding with the time: `Bienvenue sur le serveur de temps: 192.168.0.33 => 11h 26min 50s`. The Wireshark packet capture shows the network traffic between the client and the server. The first packet is a SYN packet from the client to the server, and the second packet is a SYN-ACK packet from the server to the client. The third packet is a FIN packet from the client to the server, and the fourth packet is a FIN-ACK packet from the server to the client. The terminal window also shows the connection being closed by the foreign host.

On peut déjà observer le port sur lequel la connexion s'est établie sur ordinateur, il s'agit ici du port 59952 (attribué par la licence linux), les données arrivent donc par ce port :

```
TCP 74 59952 -> 10000 [SYN] Seq=0 Win
```

On peut aussi observer des « flags » sur les trames TCP (SYN, ACK, FIN) qui correspondent au processus en cours (échange de données, connexion, déconnexion,...). Ci-dessous un schéma pour illustrer l'utilisation des flags dans une connexion client-serveur, avec :

\_SYN = demande de connexion (synchronisation)

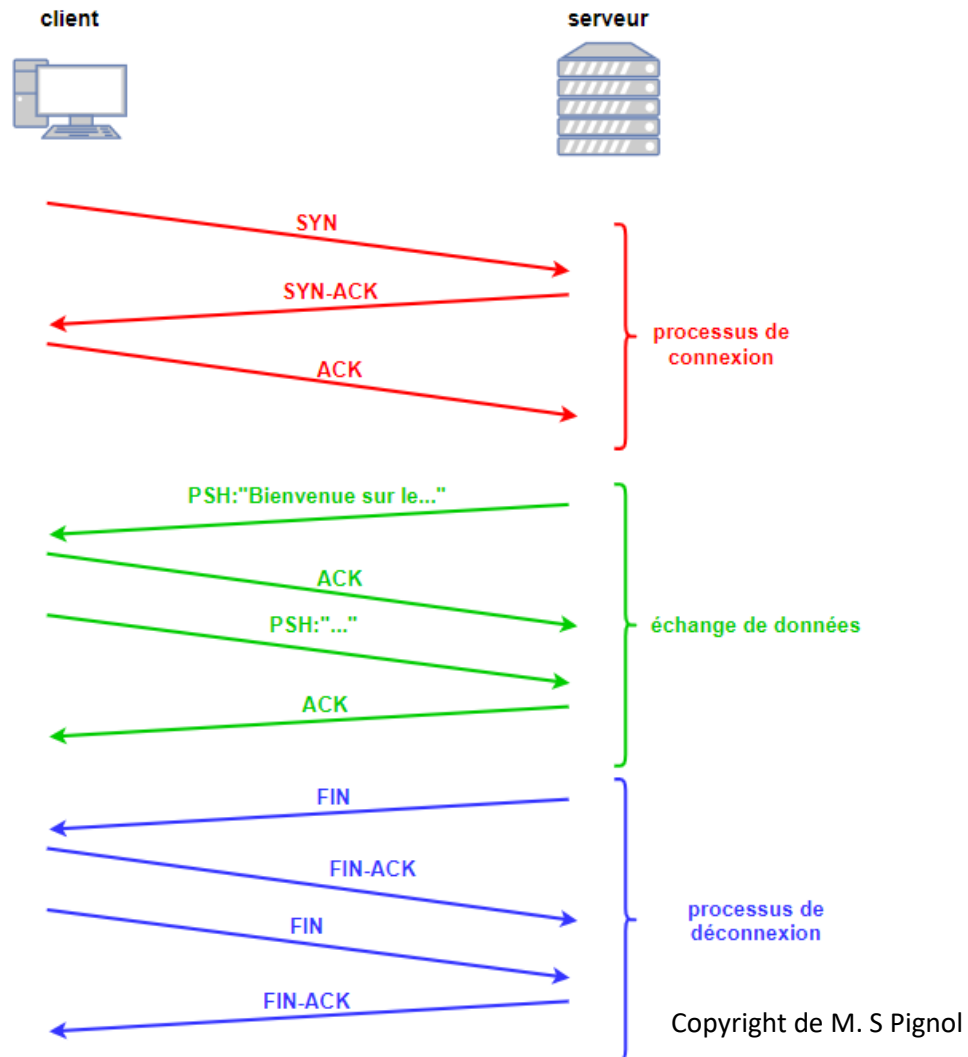
\_ACK = accusé réception

\_PSH = transfert de données

\_FIN = fin normale de connexion

Il est possible de combiner plusieurs flags dans une trame, par exemple SYN-ACK, qui demande une connexion et en même temps confirme la réception de la trame précédente :

```
10000 -> 59952 [SYN, ACK]
```



On peut donc dire que le mode TCP est un mode connecté (connexion entre les 2 machines obligatoires pour un échange de données), fiable (chaque transmission est confirmée juste après) et établit une liaison de bout en bout, car se trouvant en couche 4 du modèle OSI (couche de transport), il établit une connexion entre 2 point avec un contrôle des transmissions (flag ACK).

#### Bonus :

Si on interrompt la connexion de manière brutal (fermer le terminal, ctrl + C, débrancher le câble ethernet, etc.), on peut observer sur WireShark une trame rouge :

124	320.563419986	192.168.0.25	192.168.0.33	TCP	60 10000 → 59238 [RST, ACK] Seq=70 Ack=367 Win=64896 Len=0 TSval=3181196505 T...
125	320.563534070	192.168.0.25	192.168.0.33	TCP	60 10000 → 59238 [RST] Seq=70 Win=0 Len=0
126	325.648504707	ASISTatvC 10.4F.10	ASISTatvC 10.4F.10	APP	60 Who has 102.168.0.33? Tell 102.168.0.25

On peut aussi observer le flag RST (pour reset), qui indique une fin brutale de connexion.

#### **La commande Netstat :**

La commande *netstat* (NETwork STATistique) permet d'afficher les informations sur les connexions réseau de notre ordinateur. Par exemple, avec la commande *netstat -tln*, on affiche les connexions en attente :



```

geii@E211:~$ netstat -tln
Connexions Internet actives (seulement serveurs)
Proto Recv-Q Send-Q Adresse locale      Adresse distante     Etat
tcp      0      0 0.0.0.0:139          0.0.0.0:*            LISTEN
tcp      0      0 127.0.0.53:53        0.0.0.0:*            LISTEN
tcp      0      0 127.0.0.1:631        0.0.0.0:*            LISTEN
tcp      0      0 0.0.0.0:445          0.0.0.0:*            LISTEN
tcp6     0      0 :::139               :::*                  LISTEN
tcp6     0      0 :::80                 :::*                  LISTEN
tcp6     0      0 :::1:631              :::*                  LISTEN
tcp6     0      0 :::445                :::*                  LISTEN
geii@E211:~$ sudo service apache2 stop
geii@E211:~$ netstat -tln
Connexions Internet actives (seulement serveurs)
Proto Recv-Q Send-Q Adresse locale      Adresse distante     Etat
tcp      0      0 0.0.0.0:139          0.0.0.0:*            LISTEN
tcp      0      0 127.0.0.53:53        0.0.0.0:*            LISTEN
tcp      0      0 127.0.0.1:631        0.0.0.0:*            LISTEN
tcp      0      0 0.0.0.0:445          0.0.0.0:*            LISTEN
tcp6     0      0 :::139               :::*                  LISTEN
tcp6     0      0 :::1:631              :::*                  LISTEN
tcp6     0      0 :::445                :::*                  LISTEN
geii@E211:~$

```

On voit ainsi les ports « ouverts » tels que le 445 (partage de fichier avec le protocole smb).

L'adresse 0.0.0.0 signifie que les connexions sont possibles depuis l'extérieur de la machine.

On observe aussi que le port 80 (http) est ouvert, grâce au logiciel de serveur apache2, qui permet de créer un serveur http. On peut d'ailleurs voir que si l'on stop ce service, la machine n'est plus en attente de connexion sur le port 80 et la ligne disparaît.

A l'inverse, si je lance un serveur http sur le port 8000 avec python, on verra apparaître une attente de connexion sur le port 8000 :

```

geii@E211:~$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...

```

```

geii@E211:~$ netstat -tln
Connexions Internet actives (seulement serveurs)
Proto Recv-Q Send-Q Adresse locale      Adresse distante     Etat
tcp      0      0 0.0.0.0:139          0.0.0.0:*            LISTEN
tcp      0      0 127.0.0.53:53        0.0.0.0:*            LISTEN
tcp      0      0 127.0.0.1:631        0.0.0.0:*            LISTEN
tcp      0      0 0.0.0.0:445          0.0.0.0:*            LISTEN
tcp      0      0 0.0.0.0:8000         0.0.0.0:*            LISTEN
tcp6     0      0 :::139               :::*                  LISTEN
tcp6     0      0 :::1:631              :::*                  LISTEN
tcp6     0      0 :::445                :::*                  LISTEN

```

Tous les ports ouverts ont également l'attribut « LISTENING », car la machine « écoute » les demandes de connexion sur ces ports. Les ports ouverts sont d'ailleurs à l'origine de nombreuses failles, permettant à un pirate de se connecter à votre ordinateur.

(En 2017, le groupe de hackers *The Shadow Brokers* a révélé que le port 445 pouvait servir de porte d'entrée à votre pc grâce à la faille « *EternalBlue* » dans le protocole smb, développée par la NSA)

Avec la commande `netstat -tnp` on peut observer les connexions établies avec un serveur distant :

```
geii@E211:~$ netstat -tnp
(Tous les processus ne peuvent être identifiés, les infos sur les processus
non possédés ne seront pas affichées, vous devez être root pour les voir toutes.)
Connexions Internet actives (sans serveurs)
Proto Recv-Q Send-Q Adresse locale      Adresse distante      Etat      PID/Program name
tcp      0      0 192.168.0.33:59954    192.168.0.25:10000    ESTABLISHED 5233/telnet
geii@E211:~$
```

Dans notre cas, nous sommes toujours connectés au serveur « Chuck », on peut donc observer que la connexion est établie avec le programme telnet.

La commande `netstat -tanp` permet d'afficher toutes les connexions (établies et en attentes) sur notre ordinateur :

```
geii@E211:~$ netstat -tanp
(Tous les processus ne peuvent être identifiés, les infos sur les processus
non possédés ne seront pas affichées, vous devez être root pour les voir toutes.)
Connexions Internet actives (serveurs et établies)
Proto Recv-Q Send-Q Adresse locale      Adresse distante      Etat      PID/Program name
tcp      0      0 0.0.0.0:139          0.0.0.0:*             LISTEN    -
tcp      0      0 127.0.0.53:53         0.0.0.0:*             LISTEN    -
tcp      0      0 127.0.0.1:631         0.0.0.0:*             LISTEN    -
tcp      0      0 0.0.0.0:445           0.0.0.0:*             LISTEN    -
tcp      0      0 0.0.0.0:8000          0.0.0.0:*             LISTEN    7431/python3
tcp      0      0 192.168.0.33:59954    192.168.0.25:10000    ESTABLISHED 5233/telnet
tcp6     0      0 :::139                :::*                  LISTEN    -
tcp6     0      0 :::1:631              :::*                  LISTEN    -
tcp6     0      0 :::445                :::*                  LISTEN    -
geii@E211:~$
```

On peut utiliser la commande `netstat -rn` (r pour route et n pour numériques) pour visualiser la table de routage IP du noyau. Exemple :

```
[tihmz@parrotOS]-[~]
$netstat -rn
Table de routage IP du noyau
Destination      Passerelle      Genmask          Indic    MSS Fenêtre irtt Iface
0.0.0.0          192.168.1.10    0.0.0.0          UG        0 0        0 wlan0
172.16.78.0      0.0.0.0         255.255.255.0    U         0 0        0 vmnet1
192.168.1.0      0.0.0.0         255.255.255.0    U         0 0        0 wlan0
192.168.12.0     0.0.0.0         255.255.255.0    U         0 0        0 vmnet8
```

On peut aussi remplacer le t par un u dans les arguments de la commande pour visualiser les connexions UDP (User Datagram Protocol, protocole ne demandant pas de connexion contrairement à TCP) au lieu de TCP :

```
[tihmz@parrotOS]-[~]
$sudo netstat -uanp
Connexions Internet actives (serveurs et établies)
Proto Recv-Q Send-Q Adresse locale      Adresse distante      Etat      PID/Program name
udp      0      0 0.0.0.0:4500         0.0.0.0:*             1160/charon
udp      0      0 192.168.1.124:68     192.168.1.10:67      ESTABLISHED 781/NetworkManager
udp      0      0 0.0.0.0:500          0.0.0.0:*             1160/charon
udp6     0      0 :::4500              :::*                  1160/charon
udp6     0      0 :::500               :::*                  1160/charon
```

Enfin, la commande `netstat -ie` (i pour interface et e pour extend=plus d'informations) nous permet de visualiser les mêmes informations que la commande `ifconfig -a` (avec a pour all) .

## Le DNS :

En filtrant les trames sur UDP et le port 53, on peut visualiser les requêtes sur le service DNS.

Capture filter for selected interfaces: **udp and port 53**

En utilisant la commande `nslookup`, on peut envoyer des requêtes DNS pour obtenir l'adresse IP reliée au nom de domaine `www.meteofrance.com` par exemple.

The image shows two side-by-side screenshots. On the left is a Wireshark packet capture window with the filter 'udp and port 53'. It displays a list of packets, with packet 4 selected, showing a DNS standard query response from 192.168.1.10 to 192.168.1.124. The packet details pane shows the DNS response for 'www.meteofrance.com' with multiple IP addresses: 185.86.168.140, 185.86.168.137, 185.86.168.138, and 185.86.168.139. On the right is a Parrot Terminal window showing the execution of the command `$nslookup www.meteofrance.com`. The output displays the same IP addresses as the Wireshark packet details.

On retrouve dans la trame les adresses IP affichées dans le terminal.

On peut faire l'opération inverse, et rechercher le nom de domaine relié à l'adresse IP 145.242.11.26

The image shows two side-by-side screenshots. On the left is a Wireshark packet capture window with the filter 'udp and port 53'. It displays a list of packets, with packet 2 selected, showing a DNS standard query response from 192.168.1.10 to 192.168.1.124. The packet details pane shows the DNS response for '145.242.11.26.in-addr.arpa' with the domain name 'formuelassistanceteleprocedure.impots.gouv.fr'. On the right is a Parrot Terminal window showing the execution of the command `$nslookup 145.242.11.26`. The output displays the same domain name as the Wireshark packet details.

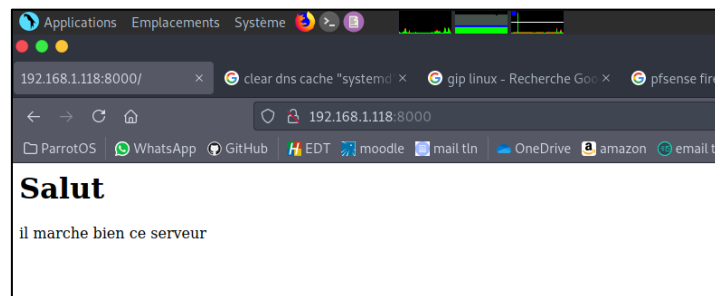
## Le protocole HTTP :

http (HyperText Transfert Protocol) est un protocole de transmission permettant d'accéder à des pages web grâce à un navigateur. HTTPS est la variante cryptée de http, beaucoup plus utilisé. On



peut aussi utiliser un terminal pour envoyer des requêtes http avec différentes commande comme *wget* par exemple.

Disclaimer : ayant oublié de prendre une capture d'écran pour la requête à l'adresse [http://stephane.pignol.univ-tln.fr/DUT\\_GEII/TP/tp\\_reseau.htm](http://stephane.pignol.univ-tln.fr/DUT_GEII/TP/tp_reseau.htm) étant accessible uniquement sur le réseau de l'université, j'ai donc refait le test avec un serveur http python sur une 2<sup>ème</sup> machine. Le serveur tournant dessus affiche simplement « Salut » et « il marche bien ce serveur » :



Après avoir vidé le contenu du cache DNS avec la commande *sudo systemd-resolve --flush-caches*

On peut ensuite réaliser une requêtes avec notre navigateur ou notre terminal avec la commande *wget adresse*. Ici, le serveur tourne sur ma machine à l'adresse 192.168.1.118 sur le port 8000 :

```
[tihmz@parrotOS]~$ wget 192.168.1.118:8000 && cat index.html
--2021-09-19 20:53:29-- http://192.168.1.118:8000/
Connexion à 192.168.1.118:8000... connecté.
requête HTTP transmise, en attente de la réponse... 200 OK
Taille : 106 [text/html]
Sauvegarde en : « index.html »

index.html      100%[=====>]      106  --.-KB/s   ds 0s

2021-09-19 20:53:29 (5,64 MB/s) – « index.html » sauvegardé [106/106]

<html>
  <body>
    <h1>Salut</h1>
    <p>il marche bien ce serveur</p>
  </body>
</html>
```

On obtient bien le code source de la page.

Maintenant, si on s'intéresse aux trames capturées par WireShark, on retrouve le même texte :

No.	Time	Source	Destination	Protocol	Length	Info
546	13.908908721	192.168.1.124	192.168.1.118	HTTP	185	GET / HTTP/1.1
582	14.108519479	192.168.1.118	192.168.1.124	HTTP	160	HTTP/1.0 200 OK (text/html)

```
Content-Length: 106\r\n
Last-Modified: Sun, 19 Sep 2021 18:50:33 GMT\r\n
\r\n
[HTTP response 1/1]
[Time since request: 0.199610758 seconds]
[Request in frame: 546]
[Request URI: http://192.168.1.118:8000/]
File Data: 106 bytes
Line-based text data: text/html (6 lines)
<html>\r\n
  <body>\r\n
    <h1>Salut</h1>\r\n
    <p>il marche bien ce serveur</p>\r\n
  </body>\r\n
</html>
```

Sur le site [http://stephane.pignol.univ-tln.fr/DUT\\_GEII/TP/tp\\_reseau.htm](http://stephane.pignol.univ-tln.fr/DUT_GEII/TP/tp_reseau.htm), la phrase affichée était

« bravo, vous avez réussi à charger la page »

On peut donc retrouver le contenu d'une page web non crypté grâce à Wireshark.

### Utilisation de GIP :

GIP est un outil d'administrateur réseau pour des calculs à partir des adresses IP, pour trouver par exemple l'étendue des adresses, le nombre d'hôtes, la longueur du préfixe pour la notation CIDR, et bien d'autres :

Calculateur de protocole Internet	
Fichier Editer Aide	
Analyseur d'adresse IPv4    Étendue IPv4 au convertisseur de préfixe    Calculateur de sous-réseau IPv4	
<b>Entrée</b>	
Adresse IP:	192.168.0.33
Masque réseau:	255.255.255.0
Longueur du préfixe:	24
<b>Sortie</b>	
Étendue des adresses:	192.168.0.0 - 192.168.0.255
Nombre de sous-réseaux:	16777216
Nombre d'hôtes:	254
Incluant Réseau/Diffusion:	256
Adresse réseau:	192.168.0.0
Adresse de diffusion:	192.168.0.255
<b>Sortie binaire</b>	
Adresse IP:	11000000.10101000.00000000.00100001
Masque réseau:	11111111.11111111.11111111.00000000
Adresse réseau:	11000000.10101000.00000000.00000000
Adresse de diffusion:	11000000.10101000.00000000.11111111
<b>Sortie hexadécimale</b>	
Adresse IP:	C0.A8.00.21
Masque réseau:	FF.FF.FF.00
Adresse réseau:	C0.A8.00.00
Adresse de diffusion:	C0.A8.00.FF

En rentrant notre adresse IP (192.168.0.33), on peut observer les informations suivantes :

Calculateur de protocole Internet	
Fichier Editer Aide	
Analyseur d'adresse IPv4    Étendue IPv4 au convertisseur de préfixe    Calculateur de sous-réseau IPv4	
<b>Entrée</b>	
Adresse IP:	192.168.0.33
Masque réseau:	255.255.255.0
Longueur du préfixe:	24
<b>Sortie</b>	
Étendue des adresses:	192.168.0.0 - 192.168.0.255
Nombre de sous-réseaux:	16777216
Nombre d'hôtes:	254
Incluant Réseau/Diffusion:	256
Adresse réseau:	192.168.0.0
Adresse de diffusion:	192.168.0.255
<b>Sortie binaire</b>	
Adresse IP:	11000000.10101000.00000000.00100001
Masque réseau:	11111111.11111111.11111111.00000000
Adresse réseau:	11000000.10101000.00000000.00000000
Adresse de diffusion:	11000000.10101000.00000000.11111111
<b>Sortie hexadécimale</b>	
Adresse IP:	C0.A8.00.21
Masque réseau:	FF.FF.FF.00
Adresse réseau:	C0.A8.00.00
Adresse de diffusion:	C0.A8.00.FF

On peut observer des informations comme l'adresse de diffusion, l'adresse réseau, le nombre d'hôtes, le nombre de sous réseau, etc.

On peut faire le même test avec une adresse tel que 192.168.50.1/22 :

Calculateur de protocole Internet

Fichier Editer Aide

Analyseur d'adresse IPv4 Étendue IPv4 au convertisseur de préfixe Calculateur de sous-réseau IPv4

**Entrée**

Adresse IP: 192.168.50.1

Masque réseau: 255.255.252.0

Longueur du préfixe: 22

**Sortie**

Étendue des adresses: 192.168.48.0 - 192.168.51.255

Nombre de sous-réseaux: 4194304

Nombre d'hôtes: 1022

Incluant Réseau/Diffusion: 1024

Adresse réseau: 192.168.48.0

Adresse de diffusion: 192.168.51.255

**Sortie binaire**

Adresse IP: 11000000.10101000.00110010.00000001

Masque réseau: 11111111.11111111.11111100.00000000

Adresse réseau: 11000000.10101000.00110000.00000000

Adresse de diffusion: 11000000.10101000.00110011.11111111

**Sortie hexadécimale**

Adresse IP: C0.A8.32.01

Masque réseau: FF.FF.FC.00

Adresse réseau: C0.A8.30.00

Adresse de diffusion: C0.A8.33.FF

On peut utiliser l'onglet calculateur de sous réseau IPv4 pour connaître le nombre de sous-reseau disponible sur une entendue d'adresses, de 192.168.1.0 à 192.168.1.255, avec les prefixe 26,27 et 28 :

Calculateur de protocole Internet

Fichier Editer Aide

Analyseur d'adresse IPv4 Étendue IPv4 au convertisseur de préfixe Calculateur de sous-réseau IPv4

Étendue des adresses: 192.168.1.0 - 192.168.1.255

Sous-réseautage utilisant la valeur de prefixlength: 26

Préfixe	Masque de sous-réseau
192.168.1.0/26	255.255.255.192
192.168.1.64/26	255.255.255.192
192.168.1.128/26	255.255.255.192
192.168.1.192/26	255.255.255.192

Calculateur de protocole Internet

Fichier Editer Aide

Analyseur d'adresse IPv4 Étendue IPv4 au convertisseur de préfixe Calculateur de sous-réseau IPv4

Étendue des adresses: 192.168.1.0 - 192.168.1.255

Sous-réseautage utilisant la valeur de prefixlength: 27

Préfixe	Masque de sous-réseau
192.168.1.0/27	255.255.255.224
192.168.1.32/27	255.255.255.224
192.168.1.64/27	255.255.255.224
192.168.1.96/27	255.255.255.224
192.168.1.128/27	255.255.255.224
192.168.1.160/27	255.255.255.224
192.168.1.192/27	255.255.255.224
192.168.1.224/27	255.255.255.224

Calculateur de protocole Internet

Fichier Editer Aide

Analyseur d'adresse IPv4 Étendue IPV4 au convertisseur de préfixe Calculateur de sous-réseau IPv4

Étendue des adresses: 192.168.1.0 - 192.168.1.255

Sous-réseauage utilisant la valeur de prefixlength: 28

Préfixe	Masque de sous-réseau
192.168.1.0/28	255.255.255.240
192.168.1.16/28	255.255.255.240
192.168.1.32/28	255.255.255.240
192.168.1.48/28	255.255.255.240
192.168.1.64/28	255.255.255.240
192.168.1.80/28	255.255.255.240
192.168.1.96/28	255.255.255.240
192.168.1.112/28	255.255.255.240
192.168.1.128/28	255.255.255.240
192.168.1.144/28	255.255.255.240
192.168.1.160/28	255.255.255.240
192.168.1.176/28	255.255.255.240
192.168.1.192/28	255.255.255.240
192.168.1.208/28	255.255.255.240
192.168.1.224/28	255.255.255.240
192.168.1.240/28	255.255.255.240

Montrant un maximum de 1000 sous-réseaux

On peut observer qu'à chaque fois qu'on augmente la taille du préfix de 1, on double le nombre de sous-réseau : 4 pour /26, 8 pour /27 et 16 pour /28. Ainsi, plus le préfix est grand, plus le nombre de sous-réseau est important.

La blague de la semaine :

