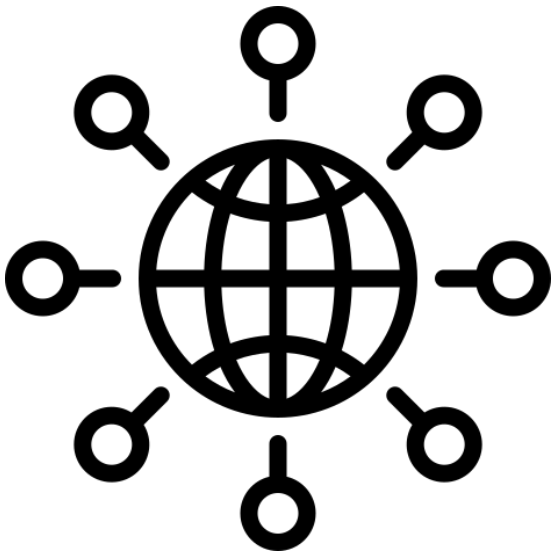
Tamda Zohir
BTS Services Informatiques aux Organisations
Bloc 1 - Mise à disposition des services informatiques

le 5 avril 2024



Les protocoles TCP et UDP

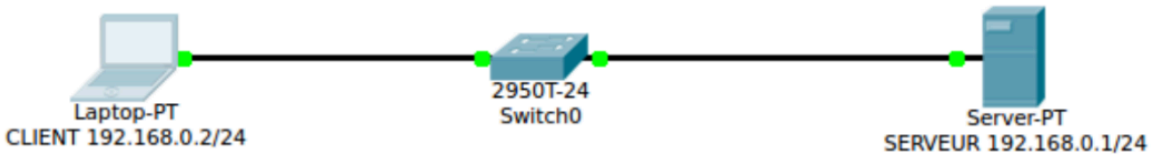
Sommaire

Définitions

Un protocole est un moyen de communication qui permet de transmettre des informations entre deux et plusieurs machines. Sur Internet, il y a actuellement deux protocoles qui sont très utilisés: **TCP** et **UDP**.

Le protocole **TCP** *(Transmission Control Protocol)* permet de transmettre des données sans aucune perte, tandis que le protocole **UDP** *(User Datagram Protocol)* va émettre des informations, peu importe les paquets qui sont perdus ou non.

Architecture du réseau



Sur ce réseau, deux machines sont connectées à un switch pour permettre de communiquer entre elles.

Dans la pratique, deux machines virtuelles sont créées grâce au logiciel VirtualBox. Ces deux dernières sont situées dans le réseau interne (c'est-à-dire isolé d'Internet) pour permettre de mieux visualiser des informations. Chaque machine a sa propre interface réseau nommée **`enp0s3`** dans laquelle il y a l'adresse IP qui est demandée sur le schéma indiqué ci-dessus.

Machine cliente:

```
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 100
0
    link/ether 08:00:27:94:2a:97 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.2/24 brd 192.168.0.255 scope global enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe94:2a97/64 scope link
       valid_lft forever preferred_lft forever
```

Machine serveur:

```
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 100
0
    link/ether 08:00:27:94:2a:97 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.1/24 brd 192.168.0.255 scope global enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe94:2a97/64 scope link dadfailed tentative
       valid_lft forever preferred_lft forever
```

Configuration réseau

Pour que les deux machines puissent communiquer entre elles, il faut configurer l'interface réseau **enp0s3** pour ces deux là. Cette configuration se situe dans le fichier **"/etc/network/interfaces"**. On peut utiliser l'éditeur de texte comme nano ou vim.

Client	Serveur

`# The primary network interface`
`auto enp0s3`
`iface enp0s3 inet static`
`address 192.168.0.2/24`
	`# The primary network interface`
	`auto enp0s3`
	`iface enp0s3 inet static`
	`address 192.168.0.1/24`

Pour prendre en compte les modifications effectuées sur ce fichier, il faut redémarrer le service réseau en tapant la commande :

```
sudo systemctl restart networking
```

TCP

Visualiser les ports ouverts

On souhaite visualiser les ports qui sont ouverts sur la machine serveur. Pour cela, il faut taper la commande “netstat -a”.

```
Connexions Internet actives (serveurs et établies)
Proto Recv-Q Send-Q Adresse locale          Adresse distante        Etat
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
```

On constate que dans la partie Adresse locale, l’adresse IP est à 0.0.0.0 suivi d’un mot “ssh”, séparés entre eux par le “:” (deux points). Cela signifie que le port entrant est ouvert sur le port 22 qui est le SSH, le reste est fermé.
Autrement dit, les autres machines peuvent se connecter en SSH à ce serveur mais ils ne peuvent pas se connecter avec d’autres protocoles comme le FTP, DNS, DHCP etc…
Dans la partie Adresse distante, l’adresse IP est à 0.0.0.0 suivi d’une étoile, séparés entre eux par un “:” (deux points). Cela signifie que toutes les machines peuvent se connecter en SSH en partant d’un port dynamique. Dans notre cas, le serveur écoute sur le port 22 qui est le SSH.

Tester la communication avec “netcat”

On souhaite s’assurer que le serveur écoute bien sur le port 5000. Pour cela, on utilise la commande netcat.

```
serveur@bts-sio:~$ netcat -l -s 192.168.0.1 -p 5000 &
[1] 566
```

Options	Description
-l	Active le mode écoute, utilise le protocole TCP par défaut.
-s	spécifier le nom d’hôte auquel on souhaite écouter (ex: 192.168.0.1).
-p	spécifier le numéro de port que l'on souhaite écouter (ex: 5000).
&	exécuter cette commande en tâche de fond (en arrière-plan).

```
tcp        0      0 localhost:5000          0.0.0.0:*               LISTEN
```

Résultat: on voit que le socket “localhost:5000” est présent dans la liste d’adresse locale. Sur l’adresse distante, “0.0.0.0:*” est indiqué dans la partie Adresse distante et l’état est en LISTEN (écoute). Ce qui signifie que le serveur est en train d’écouter toutes les machines avec le protocole TCP.

Connexion à distance avec “telnet” et “ssh”

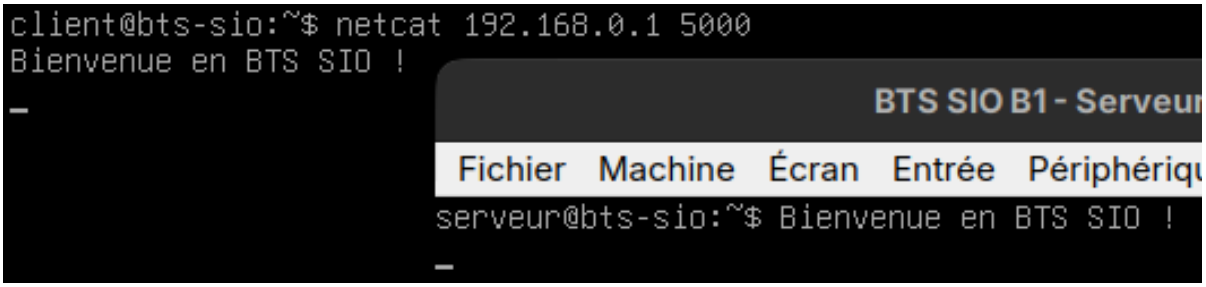
L’étape suivante consiste à se connecter au serveur à distance en utilisant le service Telnet. Côté client, il suffit de taper la commande:

Résultat: La machine cliente a réussi à se connecter à la machine serveur. Avec Telnet, on peut écrire des informations à distance.
L’inconvénient avec Telnet est la sécurité. En effet, c’est un service qui ne garantit pas le chiffrage lors de la transmission de données. Pour éviter cela, on utilise un autre service qui est le SSH (Secure Shell). Si on souhaite se connecter au serveur, il suffit de taper la commande “ssh serveur@192.168.0.1”. Cette dernière fait qu’on va se connecter en tant qu’utilisateur “serveur” via l’adresse IP du serveur.

Résultat: la connexion au serveur via SSH fonctionne avec succès.

Utilisation de la commande “netcat”

Si on souhaite transmettre des informations à la machine serveur depuis la machine cliente, la commande netcat est utilisée.



Côté serveur, on voit que l’état est en “ESTABLISHED” (établi), ce qui signifie que la machine cliente a réussi à se connecter à la machine serveur en utilisant la commande netcat.

Processus serveur multiples

On souhaite se connecter à la machine serveur à distance. On exécute plusieurs fois cette commande en arrière-plan.

```
serveur@bts-sio:~$ netcat -l -s 192.168.0.1 -p 5000 &
[1] 566
```

```
tcp        0      0 192.168.0.1:5000        0.0.0.0:*               LISTEN
tcp        0      0 192.168.0.1:5000        0.0.0.0:*               LISTEN
```

On constate que les informations sur l'adresse locale, l'adresse distante et l'état sont identiques. Cela signifie que la machine cliente a la possibilité de se connecter à distance 2 fois maximum.
On essaie de se connecter à distance depuis la machine cliente et voici ce que donne le serveur après avoir exécuté la commande netstat -a.

```
tcp        0      0 192.168.0.1:5000        0.0.0.0:*               LISTEN
tcp        0      0 192.168.0.1:5000        192.168.0.2:50174       ESTABLISHED
```

On remarque que l'un des deux processus serveur est en écoute. Ce qui fait que quand on coupe la communication avec netcat le 2ème processus est supprimé et la machine cliente pourra à nouveau se connecter, jusqu'à ce qu'il y en ait aucun pour que la connexion soit refusée.

```
Proto Recv-Q Send-Q Adresse locale          Adresse distante        Etat
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN
tcp        0      0 192.168.0.1:5000        192.168.0.2:51482       ESTABLISHED
```

1 processus restant.

```
Proto Recv-Q Send-Q Adresse locale          Adresse distante        Etat
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
```

Aucun processus, connexion impossible.

Activer une capture de trames

On souhaite analyser les trames avec l'interface réseau enp0s3. Il est possible de le faire grâce à la commande "tcpdump".

```
serveur@bts-sio:~$ sudo tcpdump -i enp0s3
```

L'option "-i" permet de spécifier l'interface réseau auquel on souhaite analyser les trames.

```
listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
14:50:16.210364 IP 192.168.0.2.52866 > 192.168.0.1.5000: Flags [S], seq 2354612477, win 64240, optio
ns [mss 1460,sackOK,TS val 153056972 ecr 0,nop,wscale 7], length 0
14:50:16.210380 IP 192.168.0.1.5000 > 192.168.0.2.52866: Flags [S.], seq 990859208, ack 2354612478,
win 65160, options [mss 1460,sackOK,TS val 507178272 ecr 153056972,nop,wscale 7], length 0
14:50:16.210507 IP 192.168.0.2.52866 > 192.168.0.1.5000: Flags [.], ack 1, win 502, options [nop,nop
,TS val 153056972 ecr 507178272], length 0
Bienvenue en BTS SIO !
                    14:50:27.619665 IP 192.168.0.2.52866 > 192.168.0.1.5000: Flags [P.], seq 1:24,
 ack 1, win 502, options [nop,nop,TS val 153068381 ecr 507178272], length 23
14:50:27.619715 IP 192.168.0.1.5000 > 192.168.0.2.52866: Flags [.], ack 24, win 509, options [nop,no
p,TS val 507189682 ecr 153068381], length 0
```

La commande "tcpdump" affiche les trames de manière verbeuse.
Chaque trame à des informations essentielles:

- La date et l'heure à laquelle la trame est effectuée.
- L'adresse et le port source (de tel adresse IP, à partir de quel port ?)
- L'adresse et le port de destination (à tel adresse IP, à quel port ?)

Envoyer la sortie de la commande avec netcat

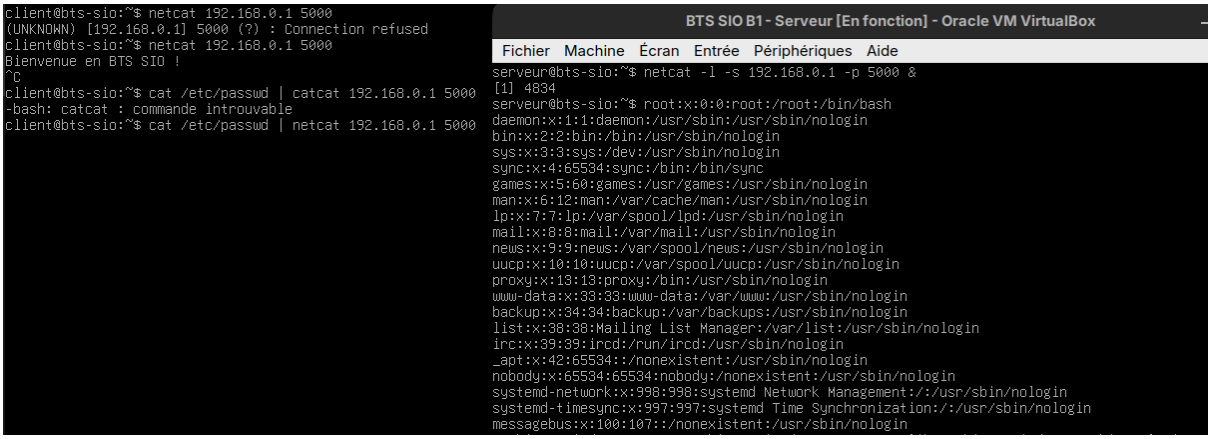
On souhaite afficher et envoyer le contenu du fichier /etc/passwd au serveur. Il faut d'abord ouvrir le port 5000.

```
serveur@bts-sio:~$ netcat -l -s 192.168.0.1 -p 5000 &
[1] 566
```

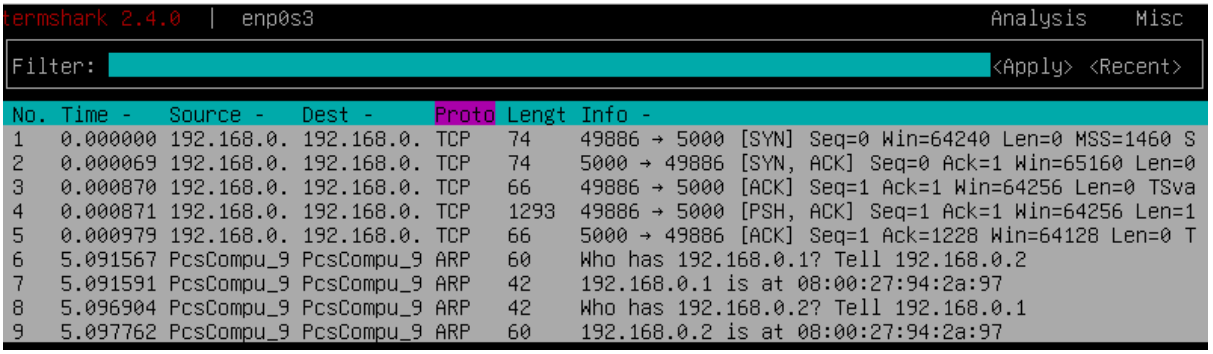
Puis, côté client, exécuter cette commande.

```
client@bts-sio:~$ cat /etc/passwd | netcat 192.168.0.1 5000
```

On remarque que sur la machine serveur, il y a le contenu du fichier /etc/passwd qui est affiché. Grâce à la commande netcat, on peut rediriger la sortie de la commande.

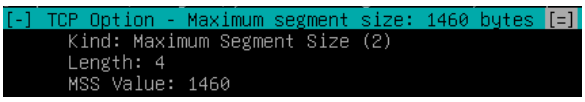


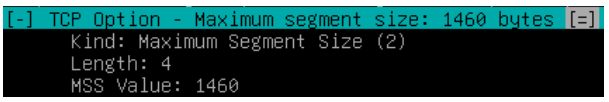
On souhaite analyser les paquets de manière visuelle et également connaître la valeur du MSS (Maximum Segment Size). Pour cela, on va utiliser le logiciel “termshark”. C’est une alternative à WireShark mais ça utilise une interface semi-graphique, adaptée pour des serveurs sans environnement de bureau. Pour cela, il suffit de taper “termshark enp0s3”.



On observe qu'il y a 9 trames qui sont capturées, dont 5 en TCP et 4 en ARP.

La partie essentielle concerne le MSS pour la 1ère et 2ème trame, celles-ci font 1460 octets et peuvent avoir une taille maximale de 1600 octets.





L’intérêt du MSS est l’optimisation des paquets. En effet, elle permet de mieux transmettre des données sans aucune fragmentation.

Détails sur les trames

On souhaite analyser les trames pour comprendre comment ils se communiquent entre eux. Pour cela, la commande “tcpdump” est utilisée.

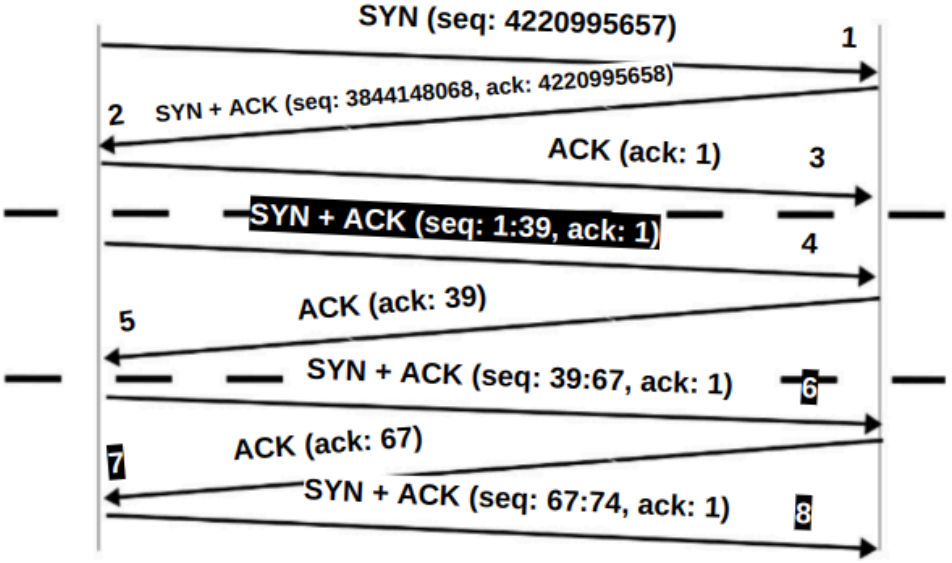
```
serveur@bts-sio:~$ sudo tcpdump -i enp0s3
```

L’option “-i” permet de spécifier l’interface réseau auquel on souhaite analyser les trames.

```
16:15:23.792775 IP 192.168.0.2.43870 > 192.168.0.1.5000: Flags [S], seq 4220995657, win 64240, optio
ns [mss 1460,sackOK,TS val 158164553 ecr 0,nop,wscale 7], length 0
16:15:23.792843 IP 192.168.0.1.5000 > 192.168.0.2.43870: Flags [S.], seq 3844148068, ack 4220995658,
 win 65160, options [mss 1460,sackOK,TS val 512285855 ecr 158164553,nop,wscale 7], length 0
16:15:23.793434 IP 192.168.0.2.43870 > 192.168.0.1.5000: Flags [.], ack 1, win 502, options [nop,nop
,TS val 158164553 ecr 512285855], length 0
```

```
CLIENT ---> hello, world ---> SERVEUR
                                16:15:36.639922 IP 192.168.0.2.43870 > 192.168.0.1.5000: Flags
[P.], seq 1:39, ack 1, win 502, options [nop,nop,TS val 158177400 ecr 512285855], length 38
16:15:36.639966 IP 192.168.0.1.5000 > 192.168.0.2.43870: Flags [.], ack 39, win 509, options [nop,no
p,TS val 512298702 ecr 158177400], length 0
```

```
CLIENT <--- OK <--- SERVEUR
                          16:15:51.112191 IP 192.168.0.2.43870 > 192.168.0.1.5000: Flags [P.], seq
39:67, ack 1, win 502, options [nop,nop,TS val 158191872 ecr 512298702], length 28
16:15:51.112240 IP 192.168.0.1.5000 > 192.168.0.2.43870: Flags [.], ack 67, win 509, options [nop,no
p,TS val 512313174 ecr 158191872], length 0
CTRL-C
```



Le nombre d’octets échangés pour la flèche n°4 est de 38 octets

- seq: 39 - 1 = 38 octets.

Le nombre d’octets échangés pour la flèche n°6 est de 35 octets

- seq: 67 - 39 = 35 octets.

Le nombre d’octets échangés pour la flèche n°8 est de 7 octets

- seq: 74 - 67 = 7 octets.

Le numéro de séquence part de 1 et va jusqu’à 74, ce qui fait un total de 73 octets échangés entre le client et le serveur.
Le numéro d’acquittement part de 1 à 67, ce qui fait un total de 66 octets échangés entre le client et le serveur.

Tester la communication avec une plage de ports

De plus, netcat propose une option qui permet à l'utilisateur de définir une plage de ports (de tel à tel port, séparés par un tiret).

On ouvre le port 5000 et on souhaite s'assurer que ce port là soit ouvert mais pas les autres ports.

```
serveur@bts-sio:~$ netcat -l -s 192.168.0.1 -p 5000 &
[1] 566
```

```
Proto Recv-Q Send-Q Adresse locale          Adresse distante        Etat
tcp        0      0 192.168.0.1:5000        0.0.0.0:*               LISTEN
```

En exécutant cette commande ci-dessous, on constate que seul le port 5000 est ouvert mais les ports allant de 5001 à 5005 sont fermés.

```
client@bts-sio:~$ netcat -vv -z 192.168.0.1 5000-5005
192.168.0.1: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.0.1] 5005 (?) : Connection refused
(UNKNOWN) [192.168.0.1] 5004 (?) : Connection refused
(UNKNOWN) [192.168.0.1] 5003 (?) : Connection refused
(UNKNOWN) [192.168.0.1] 5002 (?) : Connection refused
(UNKNOWN) [192.168.0.1] 5001 (?) : Connection refused
(UNKNOWN) [192.168.0.1] 5000 (?) open
 sent 0, rcvd 0
client@bts-sio:~$
```

Utilisation de nmap

Tester les ports avec nmap

De plus, une alternative à netcat est nmap. C'est une commande qui permet de vérifier que tels ports sont ouverts ou fermés, suivant la configuration du serveur.

```
client@bts-sio:~$ nmap 192.168.0.1
Starting Nmap 7.93 ( https://nmap.org ) at 2024-04-18 04:29 CEST
Nmap scan report for 192.168.0.1
Host is up (0.00050s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT     STATE SERVICE
22/tcp   open  ssh
5000/tcp open  upnp

Nmap done: 1 IP address (1 host up) scanned in 0.13 seconds
```

Résultat: on voit que le port 22 et 5000 sont ouverts.

Côté serveur, on voit que tous les paquets sont colorés en blanc et en rouge. En effet, nmap teste tous les ports pour permettre de savoir s'ils sont ouverts ou non. Si ce n'est pas le cas, le serveur renvoie le paquet avec [RST, ACK], signifiant que tel port est fermé; sinon il le renverrait avec [SYN, ACK].

```
termshark 2.4.0  |  enp0s3                                                    Analysis    Misc

Filter:                                                                    <Apply> <Recent>

No.  Time -    Source - Dest -    Proto Lengt Info -
1    0.000000 192.168. 192.168. TCP   74    39116 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_P
2    0.000000 192.168. 192.168. TCP   74    58328 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_
3    0.000015 192.168. 192.168. TCP   54    80 → 39116 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
4    0.000027 192.168. 192.168. TCP   54    443 → 58328 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
5    0.005474 192.168. 192.168. TCP   74    58336 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_
6    0.005474 192.168. 192.168. TCP   74    36166 → 25 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_P
7    0.005474 192.168. 192.168. TCP   74    39886 → 110 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_
8    0.005495 192.168. 192.168. TCP   54    443 → 58336 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
9    0.005519 192.168. 192.168. TCP   54    25 → 36166 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10   0.005531 192.168. 192.168. TCP   54    110 → 39886 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
```

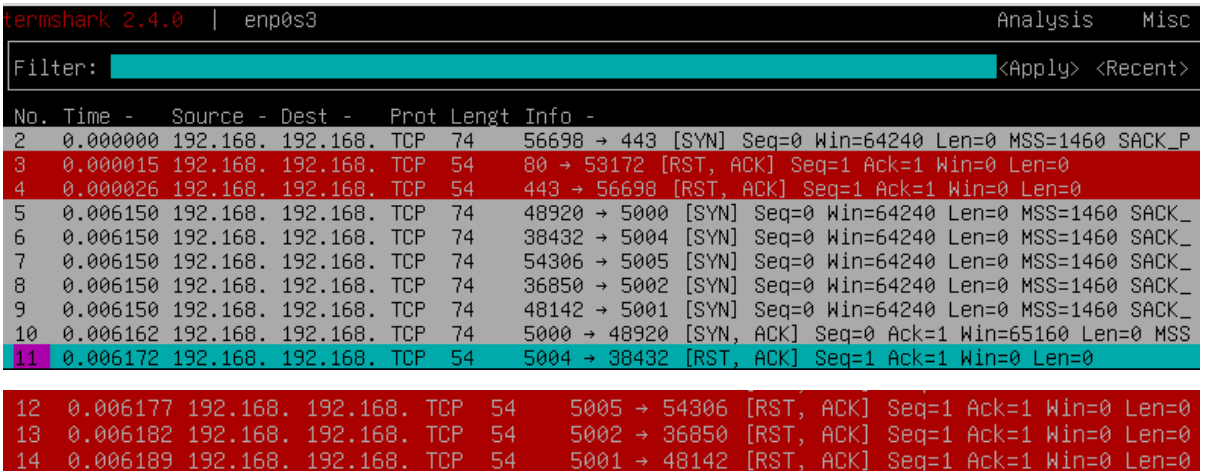
Tester la plage de ports avec nmap

Avec nmap, on a la possibilité de tester une plage de ports. Pour cela, on exécute cette commande:

```
client@bts-sio:~$ nmap -p 5000-5005 192.168.0.1
```

```
PORT     STATE  SERVICE
5000/tcp open   upnp
5001/tcp closed commplex-link
5002/tcp closed rfe
5003/tcp closed filemaker
5004/tcp closed avt-profile-1
5005/tcp closed avt-profile-2
```

Résultat: on remarque que le port 5000 est ouvert mais pas les autres ports allant de 5001 à 5005.



Sur termshark, on voit qu'il y a très peu de paquets et les ports de destination sont de 5000 à 5005. A la 10ème ligne, on constate que l'information [SYN, ACK] est indiqué, ce qui signifie que le port 5000 est ouvert et sera coloré en blanc; le reste qu'ils sont fermés seront colorés en rouge avec l'information [RST, ACK] sur termshark.

Le protocole UDP

Transmettre des données via UDP

Sur Linux, on a la possibilité d'envoyer des informations avec le protocole UDP. Pour cela, on utilise la commande "nc". L'option "-u" permet de passer en mode UDP.

```
client@bts-sio:~$ nc -vv -u 192.168.0.1 5000
192.168.0.1: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.0.1] 5000 (?) open
salut les gens
bienvenue en bts sio !
```

```
Fichier  Machine  Ecran  Entrée  Périphérique
serveur@bts-sio:~$ nc -l -p 5000 -u
salut les gens
bienvenue en bts sio !
```

Lorsqu'on a capturé des trames avec termshark (côté serveur). On voit qu'il n'y a qu'un seul paquet qui a été transmis par la machine cliente.

```
termshark 2.4.0  |  enp0s3                                                Analysis    Misc
Filter:                                                                  <Apply> <Recent>
No.  Time -       Source -          Dest -            Proto  Length  Info -
1    0.000000000  192.168.0.2       192.168.0.1       UDP    65      41722 → 5000 Len=23
2    5.010110543  PcsCompu_94:2a:9  PcsCompu_94:2a:9  ARP    60      Who has 192.168.0.1? Tell 192.16
3    5.010132760  PcsCompu_94:2a:9  PcsCompu_94:2a:9  ARP    42      192.168.0.1 is at 08:00:27:94:2a
```

```
     Destination Address: 192.168.0.1
[-] User Datagram Protocol, Src Port: 41722, Dst Port: 5000
     Source Port: 41722
     Destination Port: 5000
     Length: 31
     Checksum: 0x07bf [unverified] [=]
     Checksum Status: Unverified
     Stream index: 0
  [+] Timestamps
     UDP payload (23 bytes)
```

```
0000   08 00 27 94 2a 97 08 00  27 94 2a 97 08 00 45 00   ..'.*... '.*...E.
0010   00 33 93 91 40 00 40 11  25 d5 c0 a8 00 02 c0 a8   .3..@.@. %.......
0020   00 01 a2 fa 13 88 00 1f  07 bf 62 69 65 6e 76 65   ........ ..bienve
0030   6e 75 65 20 65 6e 20 62  74 73 20 73 69 6f 20 21   nue en b ts sio !
```

Si jamais on ouvre le port 5000 pour le protocole TCP et UDP. C'est avant tout le protocole UDP qui sera utilisé car côté client, on a spécifié l'option -u. Ce qui fait que la connexion en UDP est établie mais pas en TCP.

```
tcp        0      0 192.168.0.1:5000        0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
udp        0      0 192.168.0.1:5000        192.168.0.2:58040       ESTABLISHED
```

Conclusion

Pour en conclure, j'ai appris que les protocoles TCP et UDP ont des principes qui sont différents. En effet, le TCP transmet plusieurs paquets pour assurer la fiabilité et l'intégrité des données, tandis que l'UDP transmet qu'un seul paquet, il ne garantit pas la fiabilité mais l'optimisation lors de sa transmission d'informations.

Grâce aux outils qui sont fournis sur Linux, on a la possibilité de vérifier que les ports sont ouverts ou fermés pour permettre de garantir la sécurité du serveur (par exemple: un serveur web); mais également de s'assurer que les données sont transmises correctement avant son utilisation. Autrement dit, on fait en sorte que le pare-feu soit bien configuré côté serveur.