TP N°1 de Réseaux Assemblage et configuration d'un réseau Observations et mesures

Léo Tran Dorian Mounier Eloi Charra

14/02/2022

1 Choix des adresses

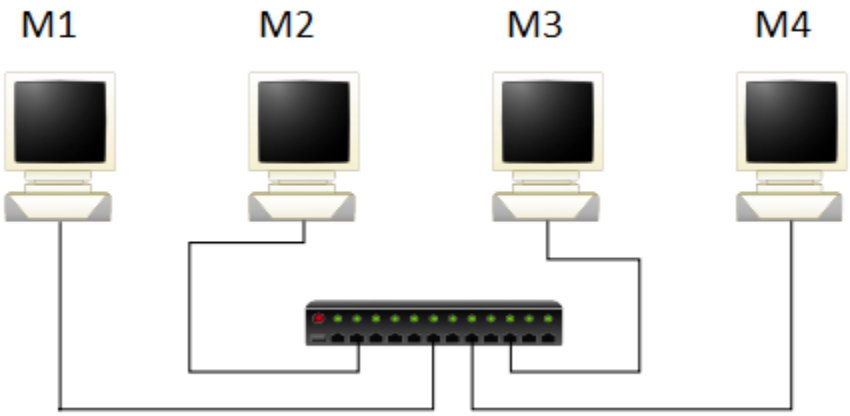

Figure 1: Schéma du réseau

Tous les ordinateurs étaient branchés au Hub avec des câbles torsadés droits.

Nous avons choisi 192.168.0.0/24 (classe C) comme adresse pour le réseau. Nous avons également fait le choix de réserver les 24 premiers bits pour identifier le réseau comme nous savions que nous allions pas utiliser beaucoup de machines. De ce fait, nous avons décidé d'allouer 8 bits (1 octet) pour la partie machine à l'aide du masque, qui nous permet de délimiter la partie réseau et la partie machine.

Les adresses des différentes machines sont les suivantes:

- M1: 192.168.0.1/24
- M2: 192.168.0.2/24
- M3: 192.168.0.3/24
- M4: 192.168.0.4/24

L'interface utilisée est bge0 qui permet de détecter les collisions. Voici comment nous avons configuré la première machine :

```
ifconfig bge0 192.168.0.1/24
```

2 Configuration des interfaces

Netmask: 0xffffff00 -> 255.255.255.0

Cela permet de nous indiquer quels octets définissent lapartie réseau et ceux qui sont destinés à identifier la machine.

Adresses de Broadcast: 192.168.0.255

Cela permet de nous indiquer quelle est l'adresse maximale du réseau, nous avons donc des adresses comprises entre la plage 192.168.0.1 et 192.168.0.254 inclues, ce qui signifie que le nombre d'adresse IP pour ce réseau est de 255.

3 Commande ping

La commande ping permet d'envoyer à intervalle régulier des paquets à la machine de destination, puis cette machine envoie une réponse ping toujours sous forme de ping. Cette commande permet de voir si deux machines sont bien connectées entre elles sur un réseau. Cette commande utilise le protocole ICMP (Internet Control Message Protocol) qui contient :

- Un type et un code (message d'erreur)
- Un checksum (validité du paquet)
- Un identifiant
- Un numéro de séquence (vérification de paquets perdus)
- des données

```
root@tpreseau:~ # ping 192.168.0.4
PING 192.168.0.4 (192.168.0.4): 56 data bytes
64 bytes from 192.168.0.4: icmp_seq=0 ttl=64 time=0.500 ms
64 bytes from 192.168.0.4: icmp_seq=1 ttl=64 time=0.601 ms
64 bytes from 192.168.0.4: icmp_seq=2 ttl=64 time=0.582 ms
^C
```

Figure 2: Aperçu de la commande ping

4084	1598.8991824…	192.168.0.4	192.168.0.2	ICMP	98	Echo (ping) reply id=0x4905, seq=326/17921, ttl=64 (request in 4083)
4085	1599.6877701…	192.168.0.1	192.168.0.4	ICMP	98	Echo (ping) request id=0xaf04, seq=351/24321, ttl=64 (reply in 4086)
4086	1599.6877782…	192.168.0.4	192.168.0.1	ICMP	98	Echo (ping) reply id=0xaf04, seq=351/24321, ttl=64 (request in 4085)
4087	1599.9620172…	192.168.0.2	192.168.0.4	ICMP	98	Echo (ping) request id=0x4905, seq=327/18177, ttl=64 (reply in 4088)
4088	1599.9620259…	192.168.0.4	192.168.0.2	ICMP	98	Echo (ping) reply id=0x4905, seq=327/18177, ttl=64 (request in 4087)
4089	1600.7506619…	192.168.0.1	192.168.0.4	ICMP	98	Echo (ping) request id=0xaf04, seq=352/24577, ttl=64 (reply in 4090)
4090	1600.7506707…	192.168.0.4	192.168.0.1	ICMP	98	Echo (ping) reply id=0xaf04, seq=352/24577, ttl=64 (request in 4089)
4091	1601.0231735…	192.168.0.2	192.168.0.4	ICMP	98	Echo (ping) request id=0x4905, seq=328/18433, ttl=64 (reply in 4092)
4092	1601.0231831…	192.168.0.4	192.168.0.2	ICMP	98	Echo (ping) reply id=0x4905, seq=328/18433, ttl=64 (request in 4091)
4093	1601.8136888…	192.168.0.1	192.168.0.4	ICMP	98	Echo (ping) request id=0xaf04, seq=353/24833, ttl=64 (reply in 4094)
4094	1601.8136973…	192.168.0.4	192.168.0.1	ICMP	98	Echo (ping) reply id=0xaf04, seq=353/24833, ttl=64 (request in 4093)
4095	1602.0867109…	192.168.0.2	192.168.0.4	ICMP	98	Echo (ping) request id=0x4905, seq=329/18689, ttl=64 (reply in 4096)
4096	1602.0867191…	192.168.0.4	192.168.0.2	ICMP	98	Echo (ping) reply id=0x4905, seq=329/18689, ttl=64 (request in 4095)
4097	1602.8463315…	192.168.0.1	192.168.0.4	ICMP	98	Echo (ping) request id=0xaf04, seq=354/25089, ttl=64 (reply in 4098)

Figure 3: Capture de Wireshark lors de la commande ping

4 Paquets ARP

La commande ping permet d'émettre des paquets de type ARP vers une adresse de destination qui n'a pas encore été utilisé. Dans ce cas-là, elle sera alors ajoutée dans ce que l'on appelle la table ARP, qui permet de stocker les résolutions MAC-IP par les ordinateurs, serveurs et éléments actifs du réseau, elle permet d'accélérer les échanges. La table ARP ne conserve pas ces adresses vers les autres machines du réseau indéfiniment, elles sont effacées au bout d'un certain temps pour ne pas encombrer la mémoire inutilement

```
root@tpreseau:~ # arp -an
? (192.168.0.2) at b4:96:91:47:9c:5c on igb0 expires in 1082 seconds [ethernet]
? (192.168.0.4) at b4:96:91:47:9c:3f on igb0 permanent [ethernet]
```

Figure 4: Table ARP de la machine M3

5 Collisions et protocole CSMA/CD

Avec la commande **netstat**, entre les machines M1 et M2, le nombre de collisions et d'erreurs sont nuls. On peut expliquer cela par le flux de données dans notre réseau qui va de la machine M1 à M2

```
root@tpreseau:~ # netstat -I igb0 10
            input          igb0           output
   packets  errs idrops      bytes    packets  errs      bytes colls
         0     0     0          0          0     0          0     0
         8     0     0        778          8     0        778     0
        14     0     0       1390         14     0       1390     0
        18     0     0       1836         18     0       1836     0
        21     0     0       2104         20     0       2040     0
        20     0     0       2040         20     0       2040     0
        19     0     0       1900         18     0       1836     0
        20     0     0       2040         20     0       2040     0
        18     0     0       1836         18     0       1836     0
        19     0     0       1938         19     0       1938     0
        14     0     0       1428         14     0       1428     0
         9     0     0        918          9     0        918     0
        11     0     0       1122         11     0       1122     0
         9     0     0        918          9     0        918     0
        10     0     0       1020         10     0       1020     0
        14     0     0       1428         14     0       1428     0
        20     0     0       2040         20     0       2040     0
        18     0     0       1836         18     0       1836     0
        18     0     0       1836         18     0       1836     0
        20     0     0       2040         20     0       2040     0
        20     0     0       2040         20     0       2040     0
```

Figure 5: Résultat de la commande netstat

Nous effectuons la commande **udpmt** entre les machines M3 et M4 pour émettre des paquets et observer des collisions sur la machine émettrice. Après 1 minute, le nombre de collisions en moyenne est de 800 collisions toutes les 10 secondes. On peut expliquer cette augmentation, par les deux trafics que nous effectuons en simultanée. Le protocole CSMA/CD permet donc de limiter le nombre de collisions.

Nous pouvons observer que plus la taille du paquet émis est grande, plus le nombre de collision est faible. La machine qui écoute sur le réseau attend la fin du paquet en cours. Si les paquets sont petits, plus il y a d'espace vide ou rien n'est reçu. Il y a alors plus de chance que la machine écoutant pense que le message est terminé. Plus les paquets sont grands, moins il y a d'espace d'attente et moins il y a de chance de se tromper.

6 Calcul de Tprop

La formule pour calculer Tprop est: $Tprop = L/V$

L = la taille du câble ETHERNET

V= Vitesse de l'onde dans le câble

Dans notre cas L vaut 1,5 mètres et V vaut $2 * 10^(8) m/s$

On a donc: $Tprop = 7,5 * 10^-9 s$

On peut donc calculer l'efficacité:

7 Débits

Le protocole CSMA/CD est utilisé lorsque nous utilisons la commande **udpmt**. Le protocole CSMA (Carrier Sense Multiple Access) permet de d détecter si un support est libre. Si c'est le cas, alors on peut lui transmettre des paquets. CD (collision detection) permet de savoir s'il y a une collision. Si c'est le cas, on arrête la transmission. Même si nous n'avons pas expérimenté de collisions, on peut penser que plus les paquets sont gros, plus la communication entre deux machines est longues. Donc une collision a plus de chance de se produire.

Voici nos mesures effectuées sur le débit par rapport à la taille du paquet envoyé :

Taille paquet	débit mesuré kbit/s
10	373
20	1413
100	3300
1000	7771
1472	7969
1473	6268
2800	8303
3000	9147

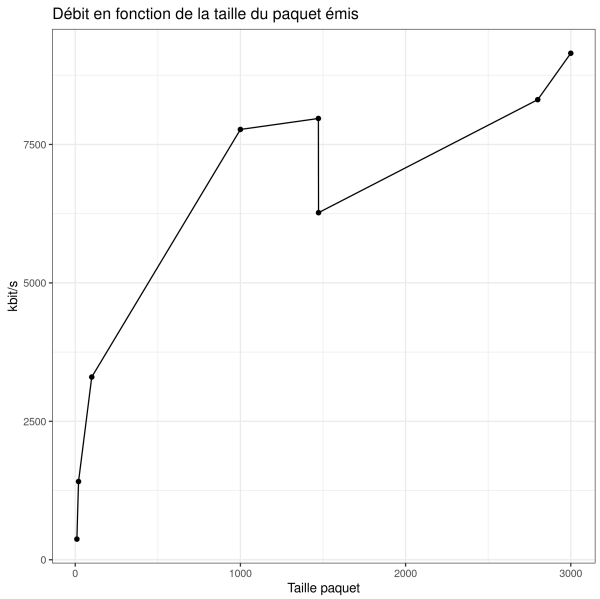

Figure 6: Courbe du débit en fonction de la taille du paquet émis

Nous pouvons remarqué un saut de la courbe entre les valeurs 1472 et 1473. Cela est du à la quantité maximale d'octets que nous pouvons émettre. Cette valeure est de 1500 octets, hors 8 octets et 20 octets sont réservés respectivement à UDP et IP. Il ne reste donc plus que 1472 octets disponibles consacrés aux données. Ce qui veut dire que les paquets ayant une taille supérieure à 1472 octets sont envoyés en plusieurs fois, d'où la diminution du débit.

8 Débit applicatif

Débit Théorique applicatif = $(tailledonnée/taillepaquet) * Débitphysique$

Ici les entêtes mesurent au total 66 octets puisque nous avons 12 octets de silence, 8 de préambule, 8 d'UDP, 20 d'IP et 18 d'Ethernet.

La taille du paquet correspond à la taille des données et la taille des entêtes.

Nous prendrons 10^7 pour le débit physique. Voici le tableau des débits applicatifs :

Taille paquet	débit mesuré kbit/s	débit applicatif kbit/s
10	373	10526
20	1413	18605
100	3300	48193
1000	7771	75046
1472	7969	76567
1473	6268	76569
2800	8303	78157
3000	9147	78278