**M2-CCI Enseignement de Réseaux – Compte-Rendu du TP n.1**

*Groupe BAJA Sara – BOLLOLI Marco*

**Introduction :**

Ce compte-rendu a pour objet les manipulations effectuées dans le TP n.1, et les observations et les calculs qui y étaient associés. De suite, les étapes seront présentées avec la numérotation adoptée dans l’énoncé du TP pour des raisons de clarté.

**Mise en place du réseau (2.1) *:***

Le réseau est mis en place à partir de 4 ordinateurs et un Hub tel que décrit dans la figure suivante.

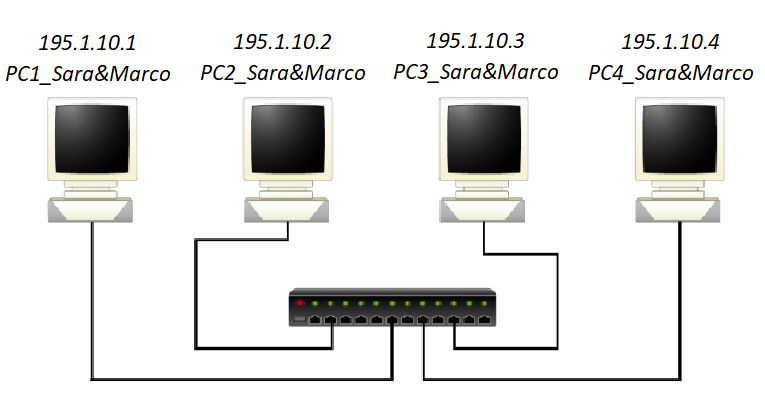


Figure 1

**Choix des adresses INTERNET des machines et configuration de l’interface :**

Dans chaque ordinateur, nous n’utilisons pas l’interface réseau de la carte mère (*em0*) mais plutôt l’interface *bge0* qui est installée en plus et qui permet d’observer des collisions entre paquets.

L’interface n’étant pas associé à une adresse IP v4, nous avons procédé à le faire pour chaque ordinateur à l’aide de la commande **ifconfig <nom interface> <adresse>**. De suite le paramètres choisis :

***Machine 1****: 195.1.10.1*

***Machine 2****:195.1.10.2*

***Machine 3****:195.1.10.3*

***Machine 4****:195.1.10.4*

Nous avons bien vérifié, à l’aide de la même commande ifconfig, que bge0 était correctement configurée sur chaque ordinateur (Cf. Figure 2).

Les informations qui nous intéressent les plus ici sont celles données par le **status** et le **media**.  Le fait que **status** est en mode active et qu’on a 10base/UTP <half-duplex> au niveau de **média** indique que l’interface est en marche.

**Contrôle de l’état des interfaces:**

Pour les quatre ordinateurs, nous observons que la valeur du **Netmask** est 0xffffff00. Cela est du au fait que nous avons utilisé une adresse IP de classe C, donc avec 24 bits de poids fort pour la partie réseau et 8 bits de poids faible pour la partie machine : donc en décimale le Netmask serait 255.255.255.0, et en hexadécimale, 0xffffff00.

Cela laisse 255 combinaisons pour la partie machine, donc 255 machines pourraient être branchées sur ce réseau. En réalité, ce sont seulement 254, car justement il y a l’adresse broadcast, qui sert, entre autres, à échanger les paquets de type ARP entre ordinateurs. Comme nous pouvons le confirmer sur la Figure 2, cet adresse est la dernière adresse machine disponible parmi les combinaisons possibles dans le réseau, donc en ce cas il s’agit de 195.1.10.255.

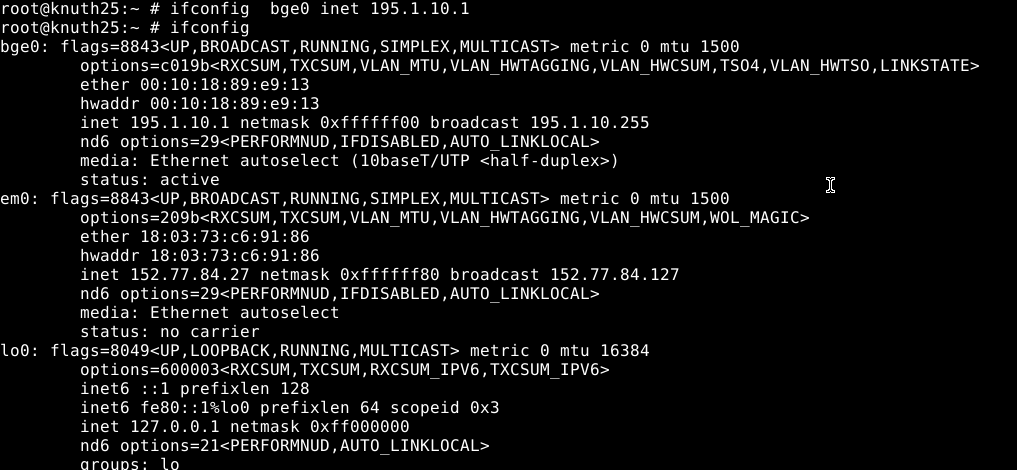


Figure 2

**Configuration par modification des fichiers de configuration:**

Une fois que nous avons vérifié le correct adressage des interfaces, les adresses sont ajoutés au fichier /etc/hosts/ de chaque ordinateur à l’aide d’un éditeur de texte, pour les faire correspondre aux noms symboliques que nous donnons aux machines au sein du réseau :

Les lignes de ce fichier sont de la forme suivante :

<Adresse Internet><nom officiel de la machine>

*195.1.10.1 – PC1\_Sara&Marco*

*195.1.10.2 – PC2\_Sara&Marco*

*195.1.10.3 – PC3\_Sara&Marco*

*195.1.10.4 – PC4\_Sara&Marco*

**Utilisation de ping :**

Par la suite, nous avons vérifié que les ordinateurs pouvaient communiquer entre eux en lançant la commande **ping <adresse>** et la commande **telnet –y <adresse>**. Nous montrons un résultat de ping à titre d’exemple en Figure 3.

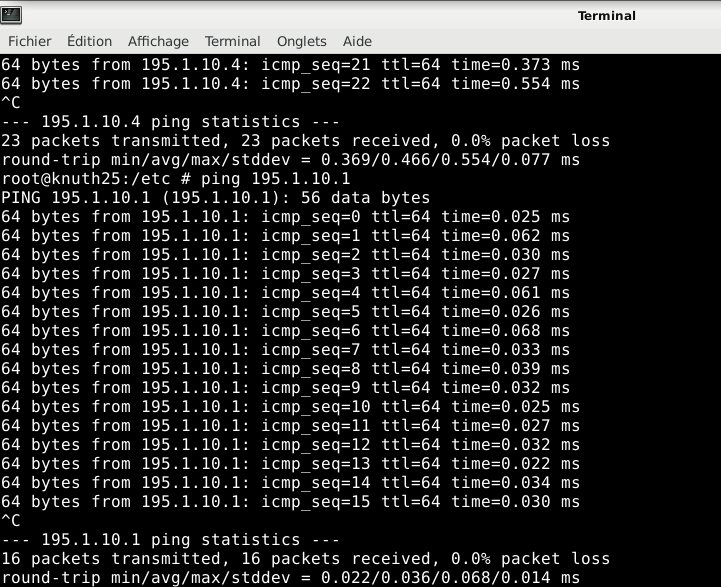


Figure 3

**Procédure de login sur une machine distante**

*Avec la commande* ***telnet****, on peut  se connecter à une machine à distance et l’utiliser via des commandes. Dans notre cas, nous avons utilisé* ***telnet –y*** *pour se connecter à la machine 2 depuis  la machine 1.*

*L’utilisation du caractère “&” s’est révélée, à posteriori, une erreur, car la Shell l’associe à d’autres opérations et donc, lors de la saisie du nom pour les commandes (par ex. pour* ***telnet –y****), cela n’était pas lu correctement, et nous avons été obligés à utiliser seulement les adresses.*

**Observation de l’activité du réseau (2.2)**

L’observation se fait à travers l’outil **Wireshark ,** c’est l’outil qui permet d’observer le réseau .Nous le lançons sur *PC2* et ensuite lançons **ping** **<195.1.10.3>**. Nous pouvons donc observer le passage de paquets de communications entre le *PC2* et le *PC3*. Ceci est montré dans la figure ce dessous.

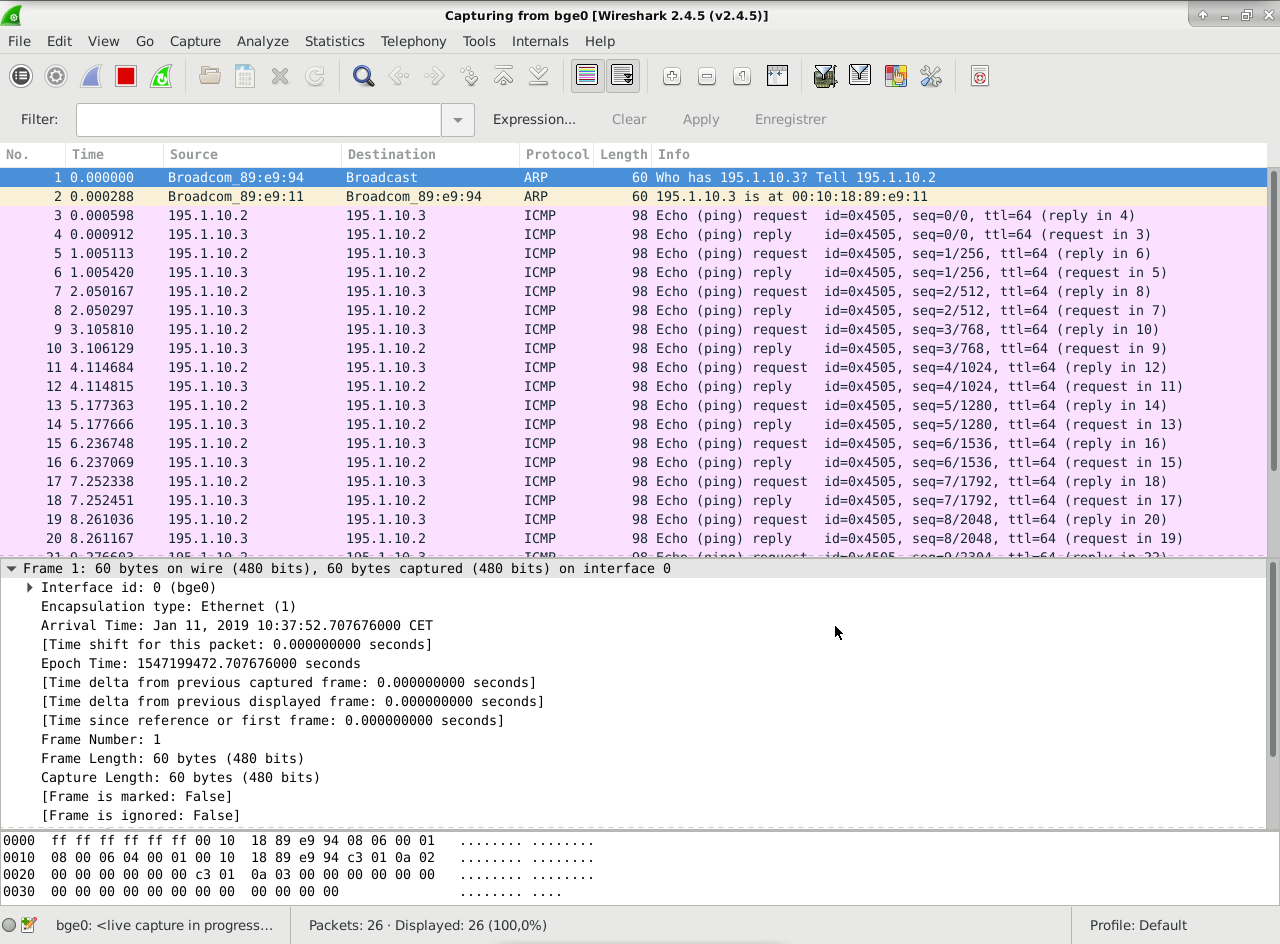


Figure 4

Cette observation nous permet de faire plusieurs remarques :

* le fonctionnement de Ping prévoit effectivement l’envoi et la réémission d’un paquet de données, de la taille minimale (64 bytes) pour Ethernet sauf si spécifié autrement.
* les paquets sont de type ICMP. Sur l’outil Wireshark nous pouvons clairement voire les différentes entêtes des paquets. D’abord il y a 14 octets de l’entête Ethernet (Figure 5) , ensuite les 20 octets du protocole IP (Figure 6), et par la suite le message ICMP (Figure 7).
* Lors du premier échange de paquets, il y a au préalable l’envoi d’un message avec le protocole ARP à travers le *broadcast*. Ce protocole sert effectivement à obtenir l’adresse de l’interface interlocutrice, et à envoyer son propre adresse. Lors des captures suivantes, nous n’avons plus observé ce couple de paquets ARP, car les interfaces connaissaient leurs adresses. Pour connaitre les adresses mémorisées, il est possible de consulter la table ARP (commande **arp –an**). Nous l’avons consulté lors de la capture, et effectivement les adresses mémorisées lors des échanges parmi les différentes machines y apparaissaient.
* Si la table ARP est vidée (**arp –d –a**), la mémoire des adresse est effacé. Dans ce cas, après délétion de la table, nous avons relancé une capture et nous avons pu observer à nouveau une paire de paquets ARP.

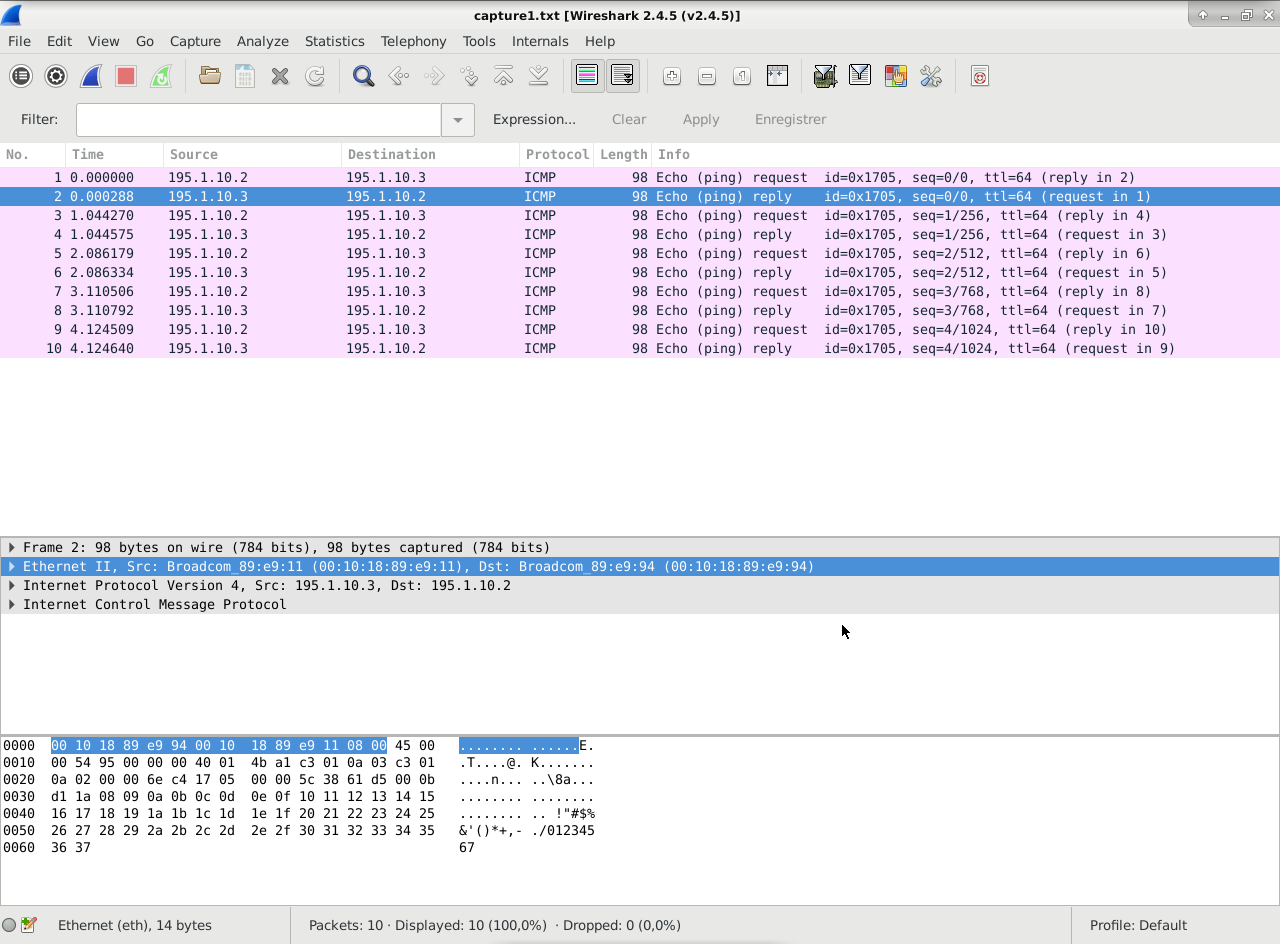


Figure 5

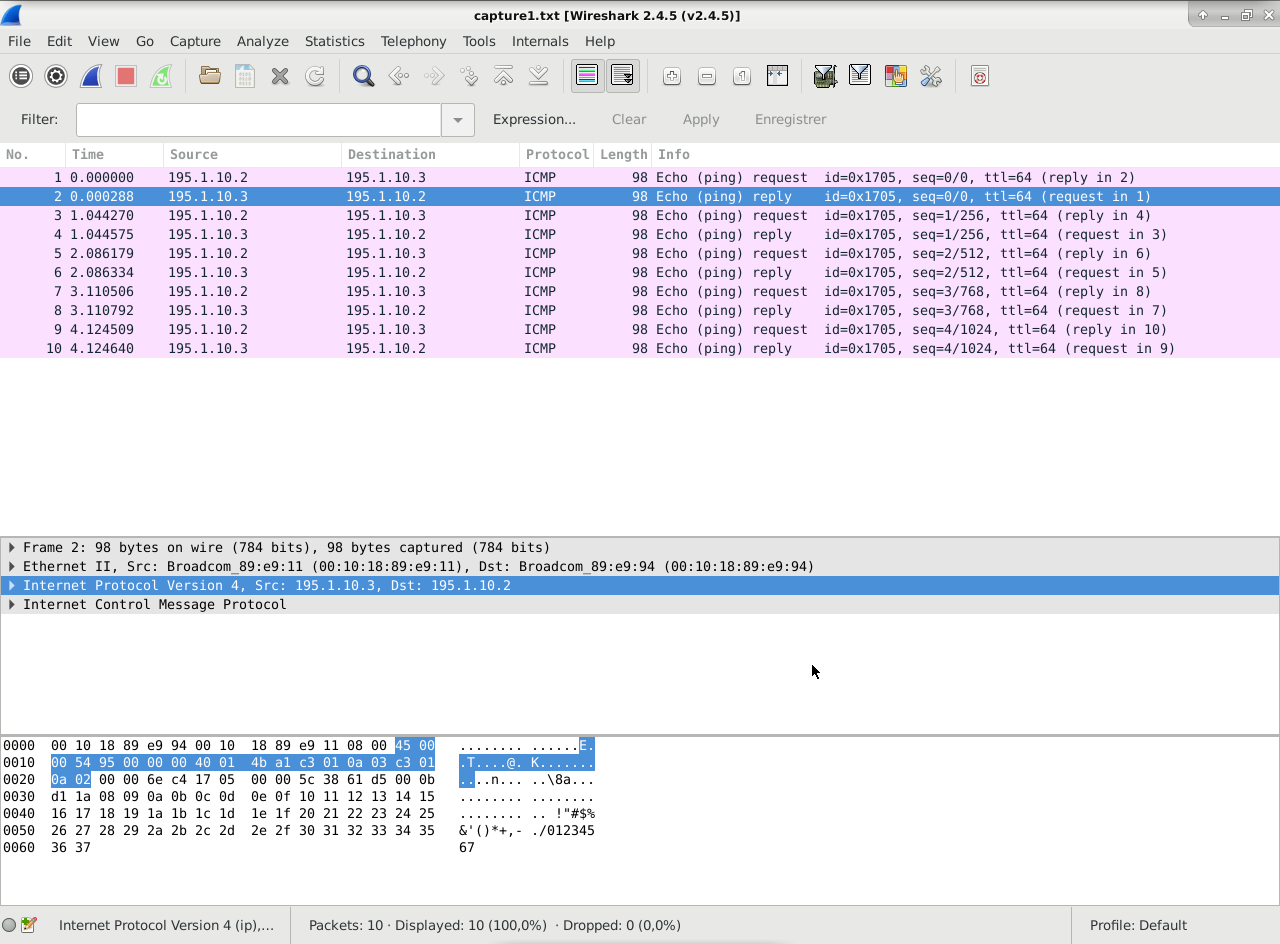


Figure 6

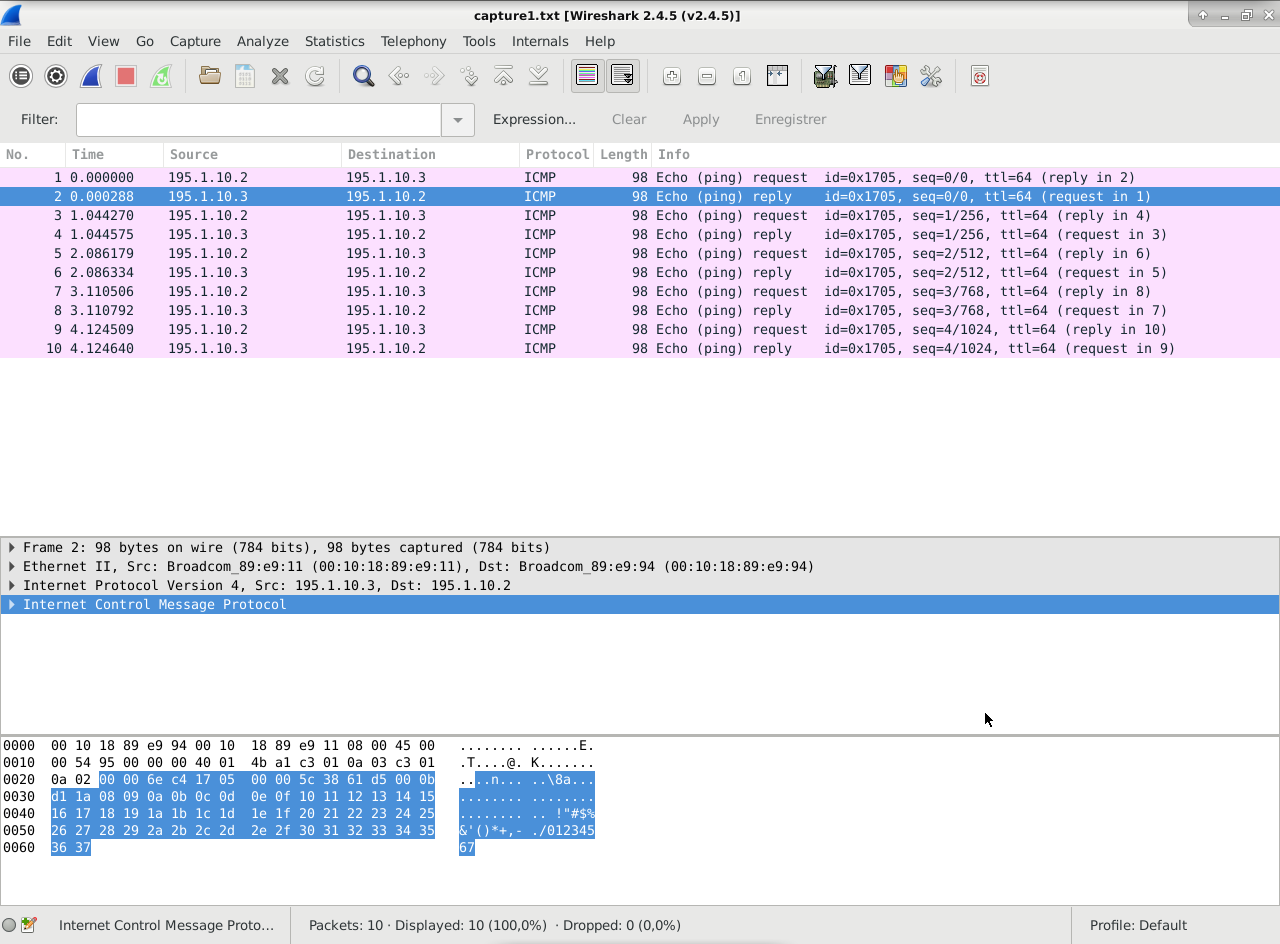


Figure 7

**Observation du protocole CSMA/CD (2.3) *:***

Nous avons utilisé l’outil Netstat pour faire des statistiques d’un transfert de paquets avec protocole UDP. Tout d’abord, nous avons fait le test entre 2 ordinateurs, *PC1* et *PC2*, *PC1* étant le client (commande **udpmt**) et *PC2* le serveur (commande **udptarget**).

Nous avons lancé d’abord la commande **netstat –I bge0 10**, pour observer le passage de paquets aux interfaces bge0 tout les 10 secondes, et ensuite les commande client-serveur.

Dans la Figure 8 nous pouvons observer les statistiques pour un envoi de paquets de 64 bytes : après une première valeur de 0, due au démarrage, le nombre de bytes et de collisions au coté client augmente jusqu’à se stabiliser entre 700 et 800. Nous n’avons pas observé d’erreurs.

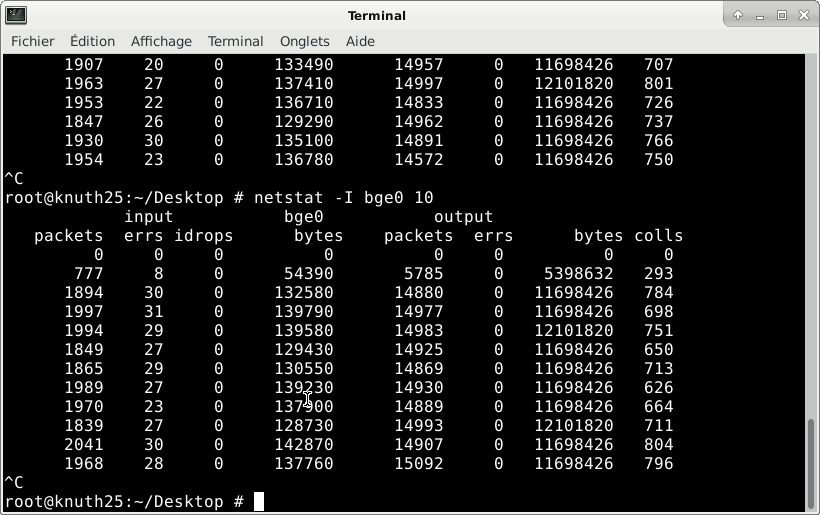


Figure 8

Si maintenant nous démarrons une deuxième connexion client-serveur entre *PC3* et *PC4*, nous pouvons observer l’impact que celle-ci a sur le nombre de collisions détectées sur *PC1* (Figure 9, centre de la page).

Effectivement nous pouvons observer que la présence de la deuxième connexion fait augmenter le nombre de collision de 700 à 1200. Le processus est, logiquement, totalement réversible, et lorsque la connexion *PC3**PC4* est terminée, nous retrouvons la valeur de 700.

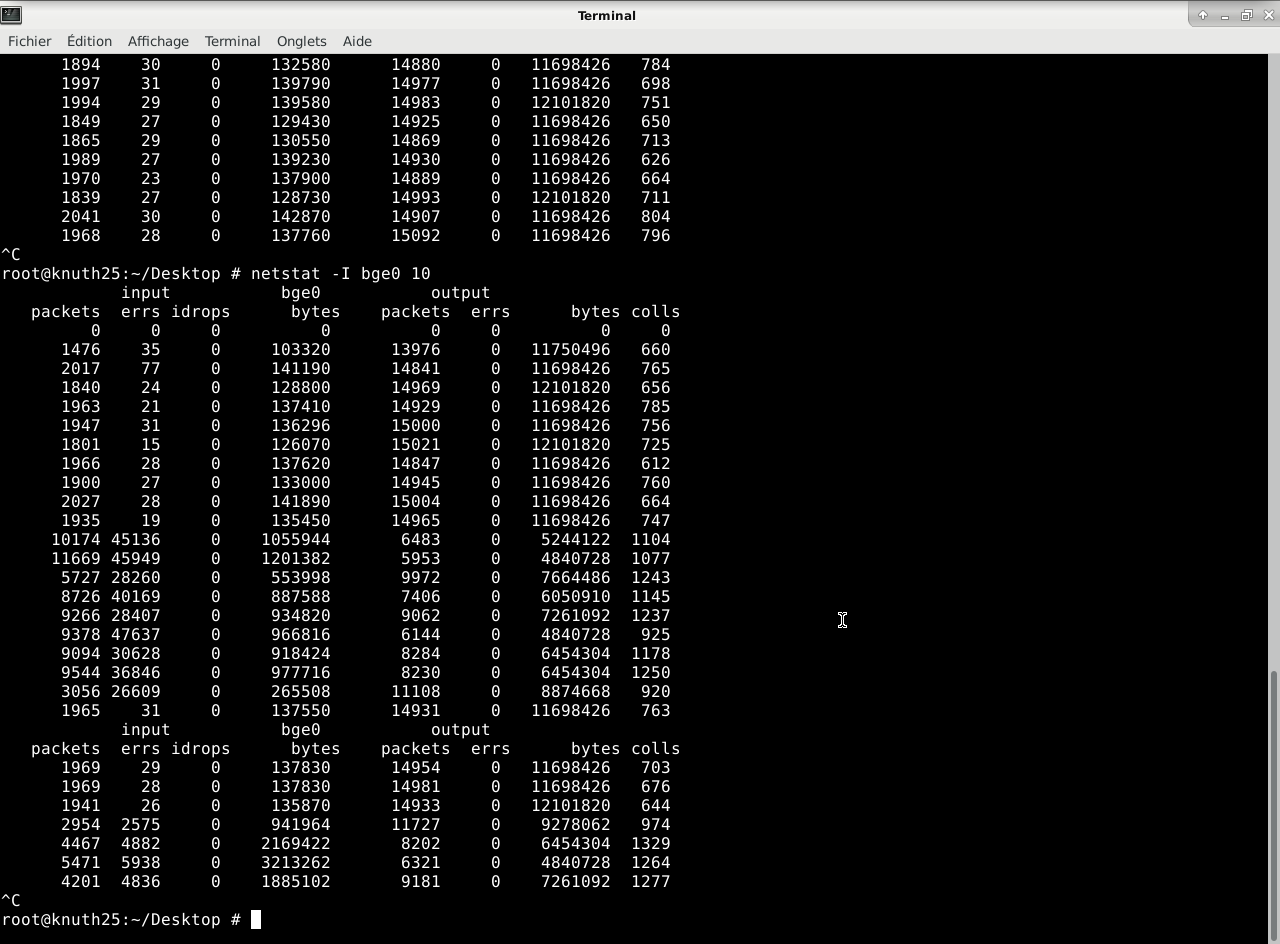


Figure 9

Ensuite, nous avons alors lancé Netstat sur tous les terminaux et étudié la variation des collisions en fonction de la taille des paquets envoyés. Un récapitulatif est fourni dans la Table 1 ci-dessous.

Nous observons que les collisions se repartissent de manière statistique entre les deux serveurs, et la même chose se produit pour les deux clients. Cela est plutôt logique, car aucune connexion n’a la priorité lors de la transmission.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| taille paquets  PC3PC4 | taille paquets PC1PC2 | collision observées à PC1 | collision observées à PC2 | collision observées à PC3 | collision observées à PC4 |
| 64 | 10 | ≈1200 | ≈1200 | ≈350 | ≈350 |
| 64 | 750 | ≈1200 | ≈1200 | ≈450 | ≈450 |
| 64 | 6000 | ≈1300 | ≈1300 | ≈700 | ≈700 |
| 6000 | 6000 | ≈1400 | ≈1400 | ≈700 | ≈700 |

Table 1

Nous aurions imaginé que, avec l’augmentation de la taille des paquets, les collisions diminuent, car il y a statistiquement moins de chances que deux paquets puissent se trouver en cours d’émission en même temps sur le réseau. Cependant, cette observation expérimentale ne va pas dans la même direction.

Nous expliquons cette différence avec le fait qu’une collision est déterminée lors du début de l’émission par le protocole CSMA/CD, ce qui arrête toute nouvelle émission, et donc un majeur nombre de collisions veut effectivement dire un majeur temps de bus pour les émissions.

Pour mieux définir la notion d’efficacité de notre réseau, nous avons calculé l’efficacité en utilisant la formule suivante :

E= 1/(1+(5,4\*Tprop/Témis)) (1)

Dans la formule (1), nous avons utilisé ces paramètres :

Tprop = longueur câbles / vitesse de l’onde dans le câble = 10 m / [(2/3) \* 3\* 108 m/s ]= 0.05 μs

Témis = taille des paquets / débit d’émission = taille paquet (bytes) / 107 bytes/s

Les résultats sont montrés en Figure 10 :

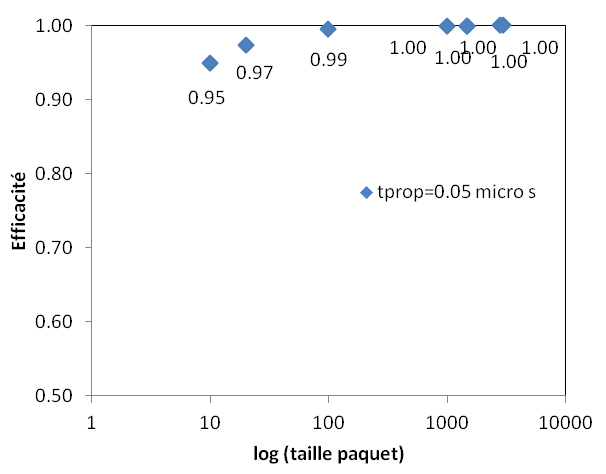


Figure 10

Du fait que notre réseau est très petit (longueur de câble très petite), l’efficacité est quasiment identique pour toute taille de paquets, ce qui explique le faible nombre de collisions détectées. Néanmoins, l’efficacité du réseau augmente en fonction de la taille des paquets. Nous pouvons conclure que dans un réseau, il est plus avantageux d’envoyer des gros paquets.

**Analyse des performances du réseau (2.4):**

Udpmt et Tcpmt permettent de calculer le débit applicatif du réseau (au niveau de l’application). Pour calculer ce débit, ils mesurent donc le temps nécessaire pour envoyer des paquets (de taille donnée) sur le réseau soit en utilisant le protocole udp soit le protocole tcp.

Calcul de débit applicatif en utilisant une seule machine:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tailles du paquet (octets)** | 10 | 20 | 100 | 1000 | 1472 | 1473 | 2800 | 3000 |
| **Débit Applicatif (Kbits/s)** | 1200 | 2300 | 6700 | 9503 | 9977 | 9036 | 9761 | 9373 |

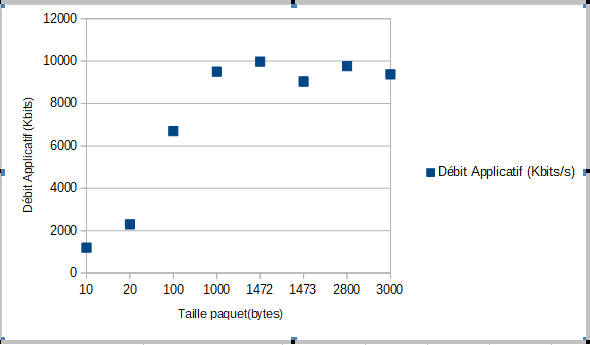


Figure 11

Nous constatons que le débit du réseau augmente en fonction de la taille des paquets. Cela montre que sur un réseau, il est plus avantageux d’envoyer des gros paquets.

Par contre, On remarque que la courbe entre les tailles 1472 et 1473 diminue et ça est due au fait …

question 11 a faire aussi ?

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tailles du paquet (octets)** | 10 | 20 | 100 | 1000 | 1472 | 1473 | 2800 | 3000 |
| **Débit Applicatif PC1(Kbits/s)** | 580 | 1129 | 2780 | 5306 | 4741 | 4462 | 5537 | 4993 |
| **Débit Applicatif PC2(Kbits/s)** | 620 | 1135 | 3232 | 4392 | 5300 | 4812 | 4703 | 4546 |

Dans le cas d’une machine émettrice pour 1000 octets, nous avons 9503 kbits/s.

Pour deux machines émettrices pour chaque 1000 octets, nous obtenons un peu prés 5000 kbits/s sur chaque machine réceptrice.

Nous remarquons que le débit est presque divisé par deux avec la configuration de deux machines émettrices. Les deux machines se partagent la trame, c’est pourquoi le débit diminue.

Test de ping de PC2 vers PC3 avec différentes taille de paquet :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tailles du paquet (octets)** | 100 | 1000 | 1472 | 1473 | 2800 | 3000 |
| **Temps-aller-retour** **(ms)** | 0.573 | 2.057 | 2.765 | 2.895 | 5.014 | 5.419 |

Nous remarquons que lors que la taille des paquets augmente le temps d’aller-retour augmente aussi. Cela s’explique par le fait que temps d’aller-retour dépend du temps d’émission des données. Plus les données sont gros, plus on a un temps d’émission qui est grand. Ainsi le temps d’aller-retour augmente aussi.

**Conclusion :**

Ce TP nous permis d’avoir des notions de base sur le fonctionnement d’un réseau Ethernet.