

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.

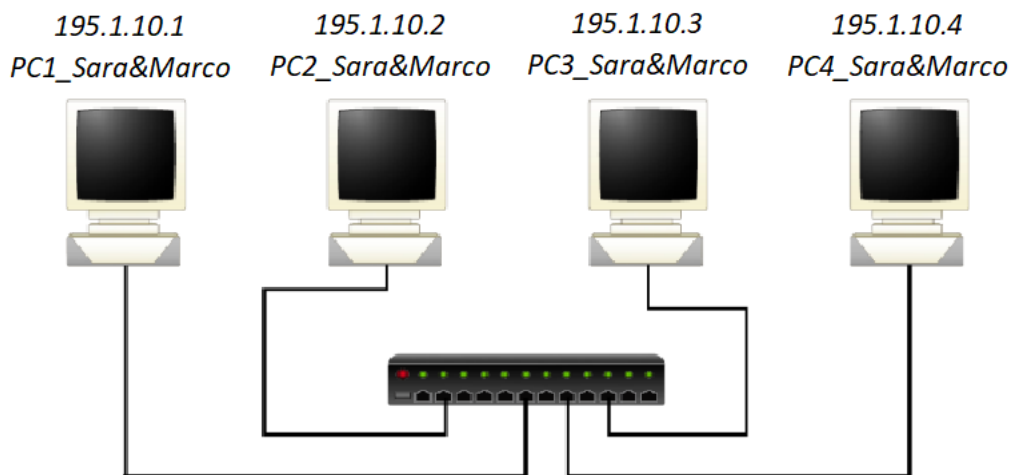


Figure 1

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 associée 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 `0xffffffff00`. Cela est dû 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, `0xffffffff00`.

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.

```
root@knuth25:~ # ifconfig bge0 inet 195.1.10.1
root@knuth25:~ # ifconfig
bge0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    options=c019b<RXCSUM, TXCSUM, VLAN_MTU, VLAN_HWTAGGING, VLAN_HWCSUM, TS04, VLAN_HWTSO, LINKSTATE>
    ether 00:10:18:89:e9:13
    hwaddr 00:10:18:89:e9:13
    inet 195.1.10.1 netmask 0xffffffff00 broadcast 195.1.10.255
    nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
    media: Ethernet autoselect (10baseT/UTP <half-duplex>)
    status: active
em0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    options=209b<RXCSUM, TXCSUM, VLAN_MTU, VLAN_HWTAGGING, VLAN_HWCSUM, WOL_MAGIC>
    ether 18:03:73:c6:91:86
    hwaddr 18:03:73:c6:91:86
    inet 152.77.84.27 netmask 0xffffffff80 broadcast 152.77.84.127
    nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
    media: Ethernet autoselect
    status: no carrier
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
    options=600003<RXCSUM, TXCSUM, RXCSUM_IPV6, TXCSUM_IPV6>
    inet6 ::1 prefixlen 128
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x3
    inet 127.0.0.1 netmask 0xff000000
    nd6 options=21<PERFORMNUD,AUTO_LINKLOCAL>
    groups: lo
```

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

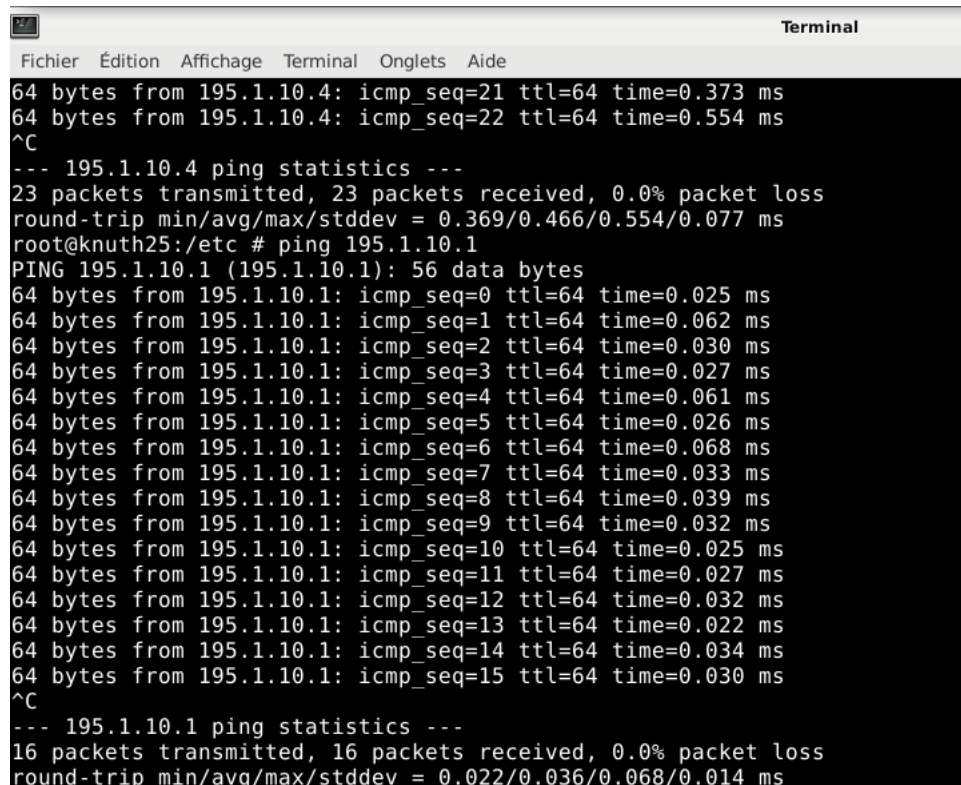
195.1.10.2 – PC2_Sara&Marco

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.



```
Terminal
Fichier  Édition  Affichage  Terminal  Onglets  Aide
64 bytes from 195.1.10.4: icmp_seq=21 ttl=64 time=0.373 ms
64 bytes from 195.1.10.4: icmp_seq=22 ttl=64 time=0.554 ms
^C
--- 195.1.10.4 ping statistics ---
23 packets transmitted, 23 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.369/0.466/0.554/0.077 ms
root@knuth25:/etc # ping 195.1.10.1
PING 195.1.10.1 (195.1.10.1): 56 data bytes
64 bytes from 195.1.10.1: icmp_seq=0 ttl=64 time=0.025 ms
64 bytes from 195.1.10.1: icmp_seq=1 ttl=64 time=0.062 ms
64 bytes from 195.1.10.1: icmp_seq=2 ttl=64 time=0.030 ms
64 bytes from 195.1.10.1: icmp_seq=3 ttl=64 time=0.027 ms
64 bytes from 195.1.10.1: icmp_seq=4 ttl=64 time=0.061 ms
64 bytes from 195.1.10.1: icmp_seq=5 ttl=64 time=0.026 ms
64 bytes from 195.1.10.1: icmp_seq=6 ttl=64 time=0.068 ms
64 bytes from 195.1.10.1: icmp_seq=7 ttl=64 time=0.033 ms
64 bytes from 195.1.10.1: icmp_seq=8 ttl=64 time=0.039 ms
64 bytes from 195.1.10.1: icmp_seq=9 ttl=64 time=0.032 ms
64 bytes from 195.1.10.1: icmp_seq=10 ttl=64 time=0.025 ms
64 bytes from 195.1.10.1: icmp_seq=11 ttl=64 time=0.027 ms
64 bytes from 195.1.10.1: icmp_seq=12 ttl=64 time=0.032 ms
64 bytes from 195.1.10.1: icmp_seq=13 ttl=64 time=0.022 ms
64 bytes from 195.1.10.1: icmp_seq=14 ttl=64 time=0.034 ms
64 bytes from 195.1.10.1: icmp_seq=15 ttl=64 time=0.030 ms
^C
--- 195.1.10.1 ping statistics ---
16 packets transmitted, 16 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.022/0.036/0.068/0.014 ms
```

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 ci-dessous.

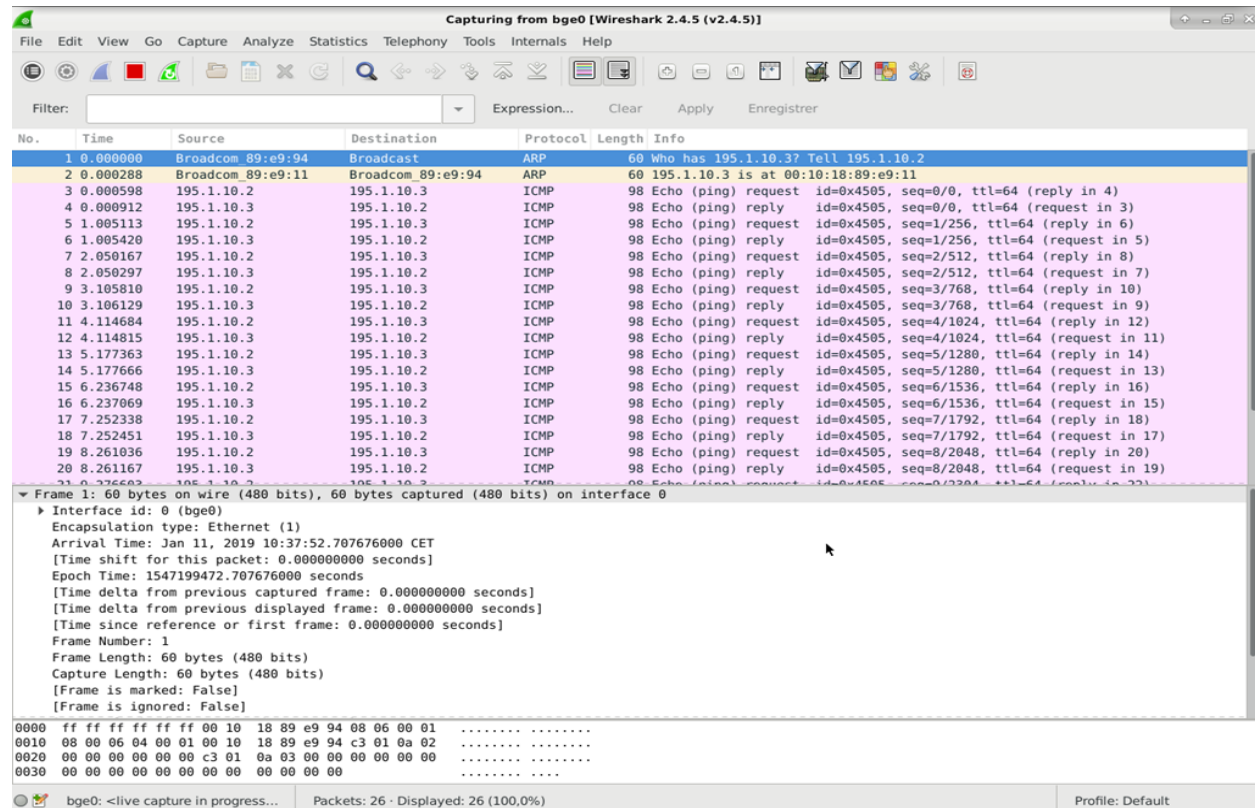


Figure 4

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 voir 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 connaissent leurs adresses. Pour connaître 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 apparaissent.
- Si la table ARP est vidée (**arp -d -a**), la mémoire des adresse est effacé. Dans ce cas, après déletion de la table, nous avons relancé une capture et nous avons pu observer à nouveau une paire de paquets ARP.

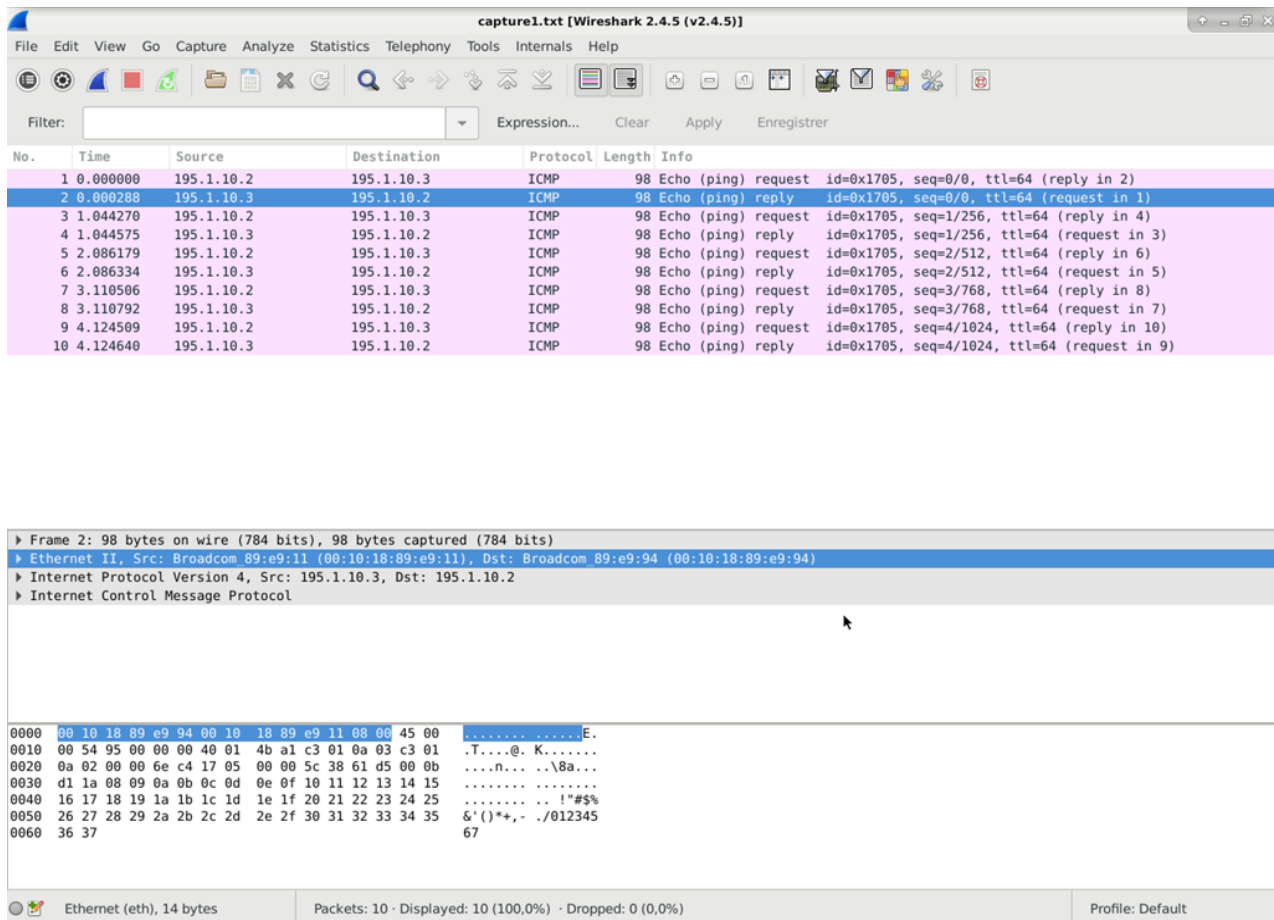


Figure 5

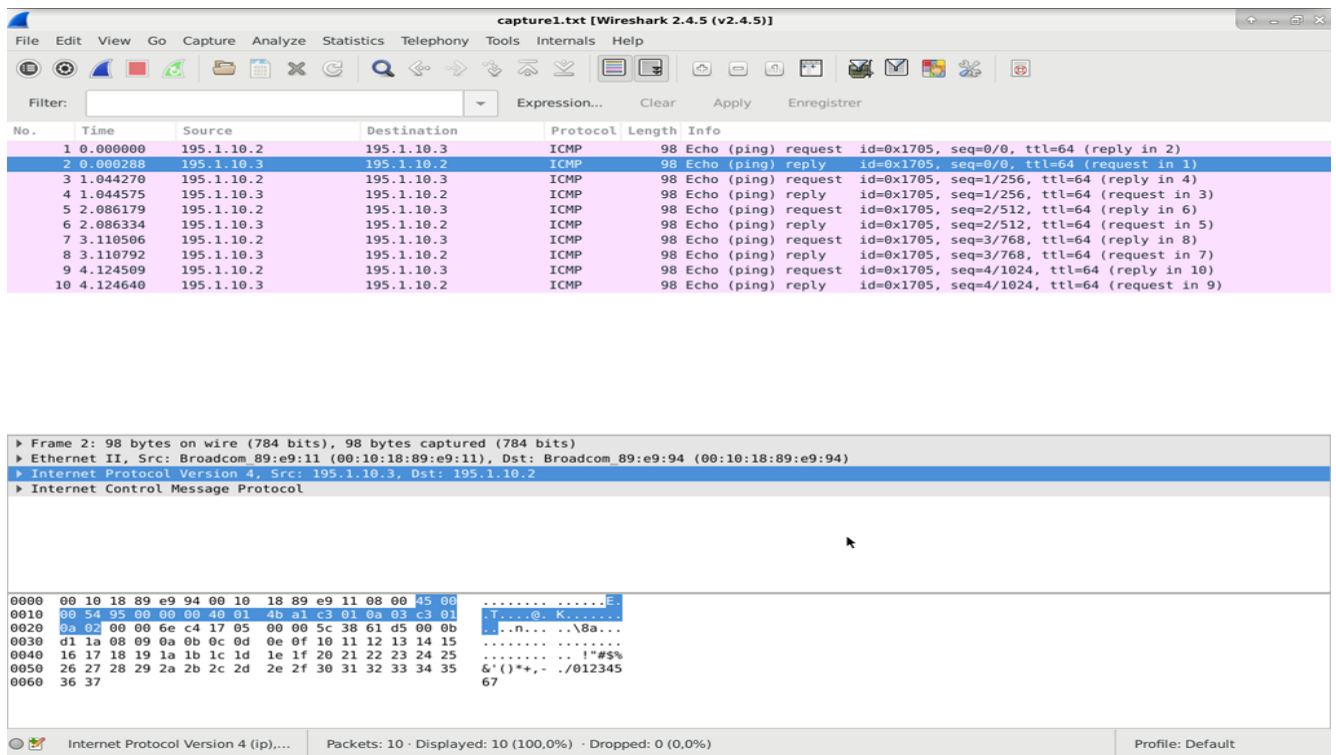


Figure 6

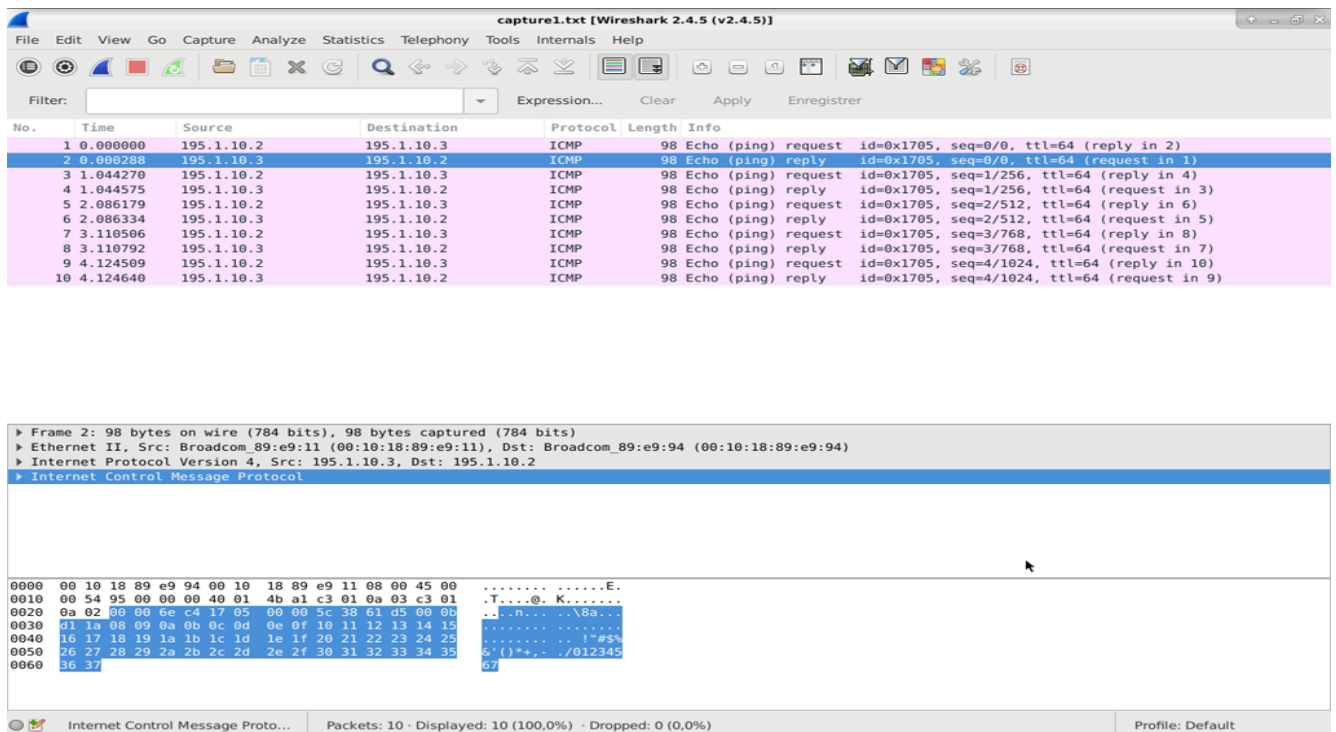


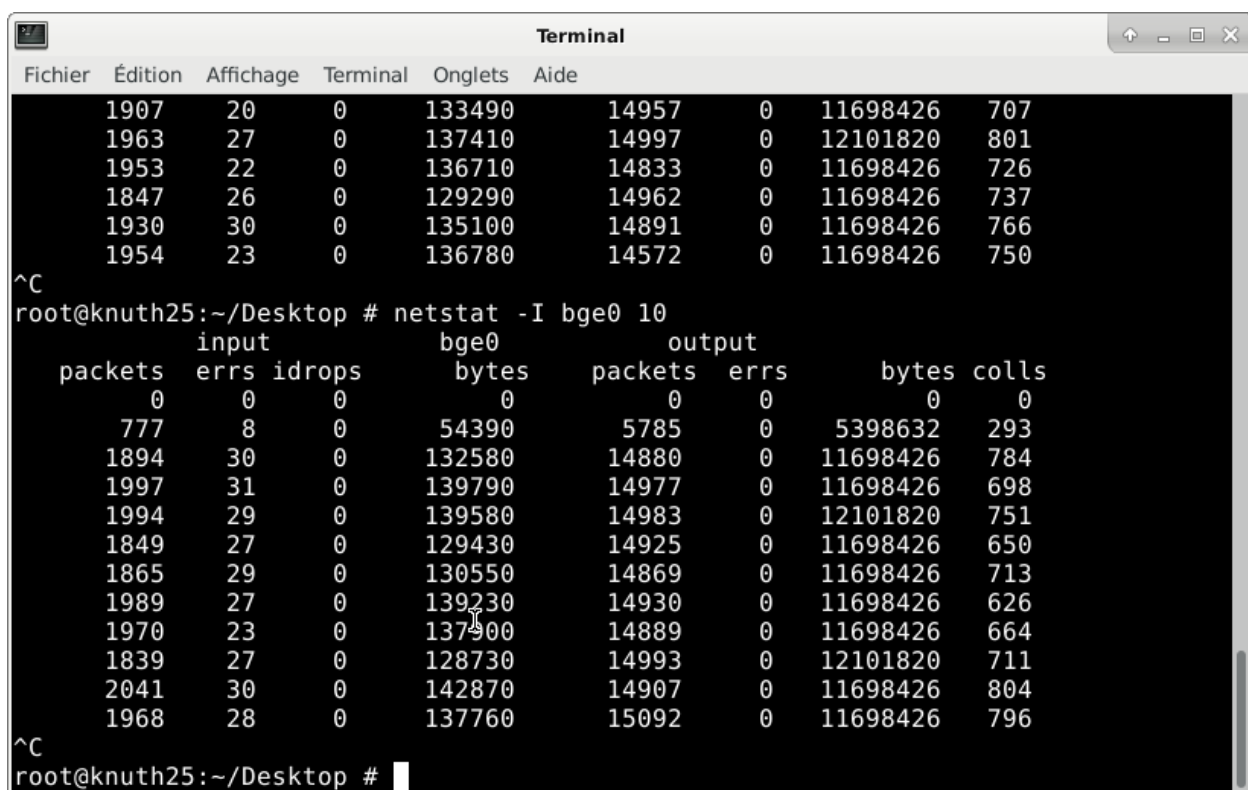
Figure 7

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.



Terminal							
Fichier	Édition	Affichage	Terminal	Onglets	Aide		
1907	20	0	133490	14957	0	11698426	707
1963	27	0	137410	14997	0	12101820	801
1953	22	0	136710	14833	0	11698426	726
1847	26	0	129290	14962	0	11698426	737
1930	30	0	135100	14891	0	11698426	766
1954	23	0	136780	14572	0	11698426	750
^C							
root@knuth25:~/Desktop # netstat -I bge0 10							
input bge0 output							
packets	errs	idrops	bytes	packets	errs	bytes	colls
0	0	0	0	0	0	0	0
777	8	0	54390	5785	0	5398632	293
1894	30	0	132580	14880	0	11698426	784
1997	31	0	139790	14977	0	11698426	698
1994	29	0	139580	14983	0	12101820	751
1849	27	0	129430	14925	0	11698426	650
1865	29	0	130550	14869	0	11698426	713
1989	27	0	139230	14930	0	11698426	626
1970	23	0	137500	14889	0	11698426	664
1839	27	0	128730	14993	0	12101820	711
2041	30	0	142870	14907	0	11698426	804
1968	28	0	137760	15092	0	11698426	796
^C							
root@knuth25:~/Desktop #							

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.


```

Terminal
Fichier  Edition  Affichage  Terminal  Onglets  Aide

1894 30 0 132580 14880 0 11698426 784
1997 31 0 139790 14977 0 11698426 698
1994 29 0 139580 14983 0 12101820 751
1849 27 0 129430 14925 0 11698426 650
1865 29 0 130550 14869 0 11698426 713
1989 27 0 139230 14930 0 11698426 626
1970 23 0 137900 14889 0 11698426 664
1839 27 0 128730 14993 0 12101820 711
2041 30 0 142870 14907 0 11698426 804
1968 28 0 137760 15092 0 11698426 796

^C
root@knuth25:~/Desktop # netstat -I bge0 10
      input      bge0      output
 packets  errs idrops  bytes  packets  errs      bytes  colls
    0      0      0      0      0      0      0      0
  1476    35      0   103320   13976      0  11750496   660
 2017    77      0   141190   14841      0  11698426   765
 1840    24      0   128800   14969      0  12101820   656
 1963    21      0   137410   14929      0  11698426   785
 1947    31      0   136296   15000      0  11698426   756
 1801    15      0   126070   15021      0  12101820   725
 1966    28      0   137620   14847      0  11698426   612
 1900    27      0   133000   14945      0  11698426   760
 2027    28      0   141890   15004      0  11698426   664
 1935    19      0   135450   14965      0  11698426   747
10174 45136      0  1055944   6483      0   5244122  1104
11660 45949      0  1201382   5953      0   4840728  1077
 5727 28260      0   553998   9972      0   7664486  1243
 8726 40169      0   887588   7406      0   6050910  1145
 9266 28407      0   934820   9062      0   7261092  1237
 9378 47637      0   966816   6144      0   4840728   925
 9094 30628      0   918424   8284      0   6454304  1178
 9544 36846      0   977716   8230      0   6454304  1250
 3056 26609      0   265508   11108      0   8874668   920
 1965    31      0   137550   14931      0  11698426   763

      input      bge0      output
 packets  errs idrops  bytes  packets  errs      bytes  colls
  1969    29      0   137830   14954      0  11698426   703
 1969    28      0   137830   14981      0  11698426   676
 1941    26      0   135870   14933      0  12101820   644
 2954 2575      0   941964   11727      0   9278062   974
 4467 4882      0  2169422   8202      0   6454304  1329
 5471 5938      0  3213262   6321      0   4840728  1264
 4201 4836      0  1885102   9181      0   7261092  1277

^C
root@knuth25:~/Desktop #

```

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.

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	0	0
64	750	≈1200	≈1200	0	0
64	6000	≈1300	≈1300	0	0
6000	6000	≈1400	≈1400	0	0

Table 1

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. Aussi les collisions ne sont observées qu'aux

interfaces émettrices : cela aussi est logique, car les interfaces receptrices ne testent pas le réseau à la recherche des collisions.

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 :

$$E = 1 / (1 + (5,4 * T_{prop} / T_{émis})) \quad (1)$$

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

T_{prop} = longueur câbles / vitesse de l'onde dans le câble = 10 m / $[(2/3) * 3 * 10^8 \text{ m/s}] = 0.05 \mu\text{s}$

$T_{émis}$ = taille des paquets / débit d'émission = taille paquet (bytes) / 10^7 bytes/s

Les résultats sont montrés en Figure 10 :

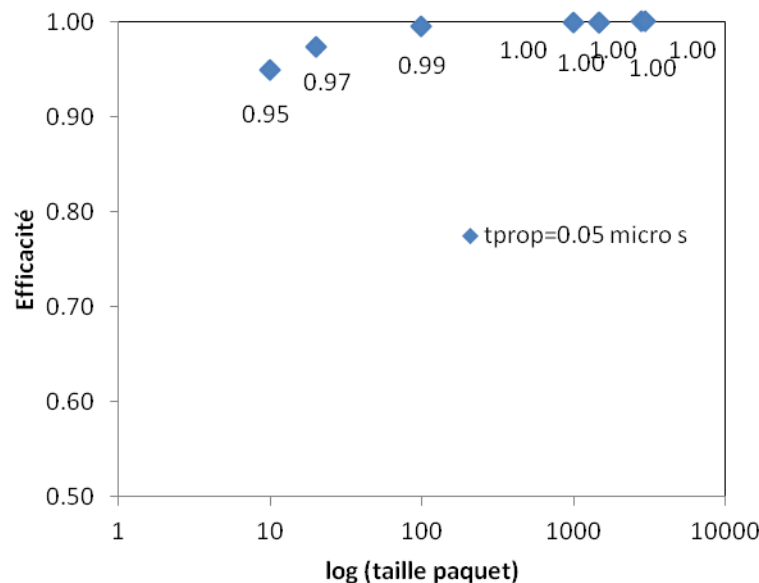


Figure 10

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.

Voici la masuration du 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

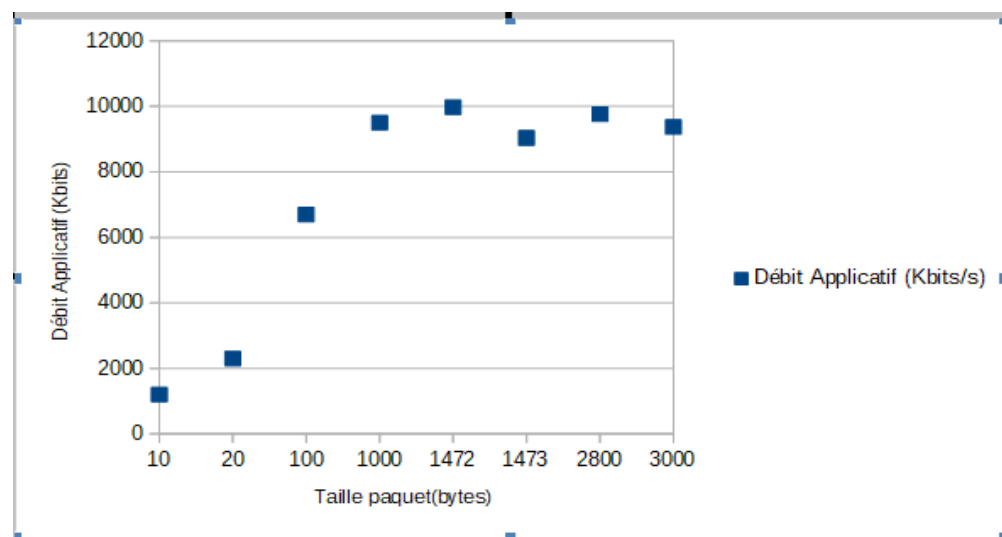


Figure 11

Nous constatons que le débit du réseau augmente en fonction de la taille des paquets. Cela est logique car les en-têtes ont un poids de plus en plus significatif dès lors que la taille des données diminue. Comme demandé dans la tâche 11, nous avons réalisé un calcul théorique du débit applicatif à partir de la formule suivant :

$$\text{Débit applicatif} = \text{Débit physique} * \text{taille données/taille paquet} \quad (2)$$

Avec Débit physique = 10 Mbits/s.

Tailles du paquet (octets)	10	20	100	1000	1472	1473	2800	3000
Débit Applicatif Mesuré (Kbits/s)	1200	2300	6700	9503	9977	9036	9761	9373
taille données/taille paquet	0.16	0.30	0.68	0.96	0.97	0.97	0.98	0.98
Débit Applicatif Théorique (Kbits/s)	1563	3030	6849	9560	9697	9697	9838	9849

Table 2

Nous remarquons tout de même que la courbe entre les tailles 1472 et 1473 diminue ; cela est dû au fait que, étant 1500 octets la taille maximale de l'ensemble des données et de l'en-tête, et étant 28 octets la taille de l'en-tête IP+UDP, si la taille des données seule est de 1473 octets alors le protocole Ethernet devra diviser l'envoi en 2 sous-paquets, ce qui a un impact sur le débit.

Nous avons ensuite testé le débit applicatif pour 2 connexions qui se font en même temps, avec la même configuration vue par avance (PC1→PC2, PC→PC4). Les résultats sont reportés ci-dessous .

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

Table 3

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 près 5000 kbits/s sur chaque machine réceptrice.

Nous remarquons que le débit est presque divisé par deux avec la configuration de deux machines émettrices : logiquement les deux machines se partagent la trame de manière égalitaire, ce qui amène à ce résultat

Mesure de Latence (2.4.3):

Pour mesurer le temps de Latence, c'est-à-dire le temps que les données passent dans les couches protocolaires, nous avons relancé un test est de ping de PC2 vers PC3, avec différentes taille de paquet. Le temps expérimental d'aller-retour nous a servi pour calculer le demi-trajet, t nous avons ensuite soustrait le $T_{\text{émis}}$ et le T_{prop} pour obtenir le temps de latence.

Nous avons considéré 2 possibilités : un calcul sans considérer le découpage en paquets de 1500 bytes effectué par le protocole Ethernet, et une approche qui tiendrait compte de ce découpage, ce qui nous semble plus correcte.

Aussi, nous avons ignoré les tests à 10 et 20 bytes de taille de paquets, car les paquets trop petit pour l'outil pour nous fournir un $T_{\text{aller-retour}}$.

Les résultats sont reportés ci-dessous :

Tailles du paquet (octets)	100	1000	1472	1473	2800	3000
$T_{\text{aller-retour}}$ (ms)	0.573	2.057	2.765	2.895	5.014	5.419
T_{aller} (ms)	0.287	1.029	1.383	1.448	2.507	2.710
$T_{\text{émis}}$ (ms)	0.010	0.100	0.147	0.147	0.280	0.300
$T_{\text{émis}}$ bis (ms) <i>si on considère le découpage des paquets</i>	0.010	0.100	0.147	0.295	0.560	0.600
T_{prop} (ms)	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}
T_{latence} (ms)	0.276	0.928	1.235	1.300	2.227	2.409
T_{latence} bis (ms)	0.276	0.928	1.235	1.153	1.947	2.109
$T_{\text{latence}}/T_{\text{aller}}$ (%)	96.5%	90.3%	89.4%	89.8%	88.8%	88.9%
$T_{\text{latence}}/T_{\text{aller}}$ bis (%)	96.5%	90.3%	89.4%	79.6%	77.7%	77.9%

Table 4

Nous remarquons que lors que la taille des paquets augmente le temps d'aller-retour augmente aussi. Nous observons clairement que cela est lié à une augmentation du poids relatif du $T_{\text{émis}}$ sur le total, et que le poids relatif du temps de latence, bien qu'il reste la composante dominante, décroît avec l'augmentation de la taille. Cela nous semble logique car plus les données sont volumineuses, plus on a un temps d'émission et un temps d'attente, en cas de collisions, qui sont grands.

Conclusion :

Ce TP nous permis d'avoir des notions de base sur le fonctionnement d'un réseau Ethernet. Nous avons pu constater quelles sont les différentes composantes du temps de transmission des données d'une machine à une autre, et nous avons-nous initier à la gestion d'une réseau avec plusieurs machines connectées ensemble.