M2PCCI

Cheick CISSOKO

Gaëtan LAGIER

Eric THIERRY

RAPPORT TP3

CONFIGURATION DES MACHINES POUR LE PROTOCOLE UDP :

Machine 1 :

Adresse IP : 192.168.0.1 (PC1) ;

Port : 59225 ;

Id : 3 ;

Machine 2 :

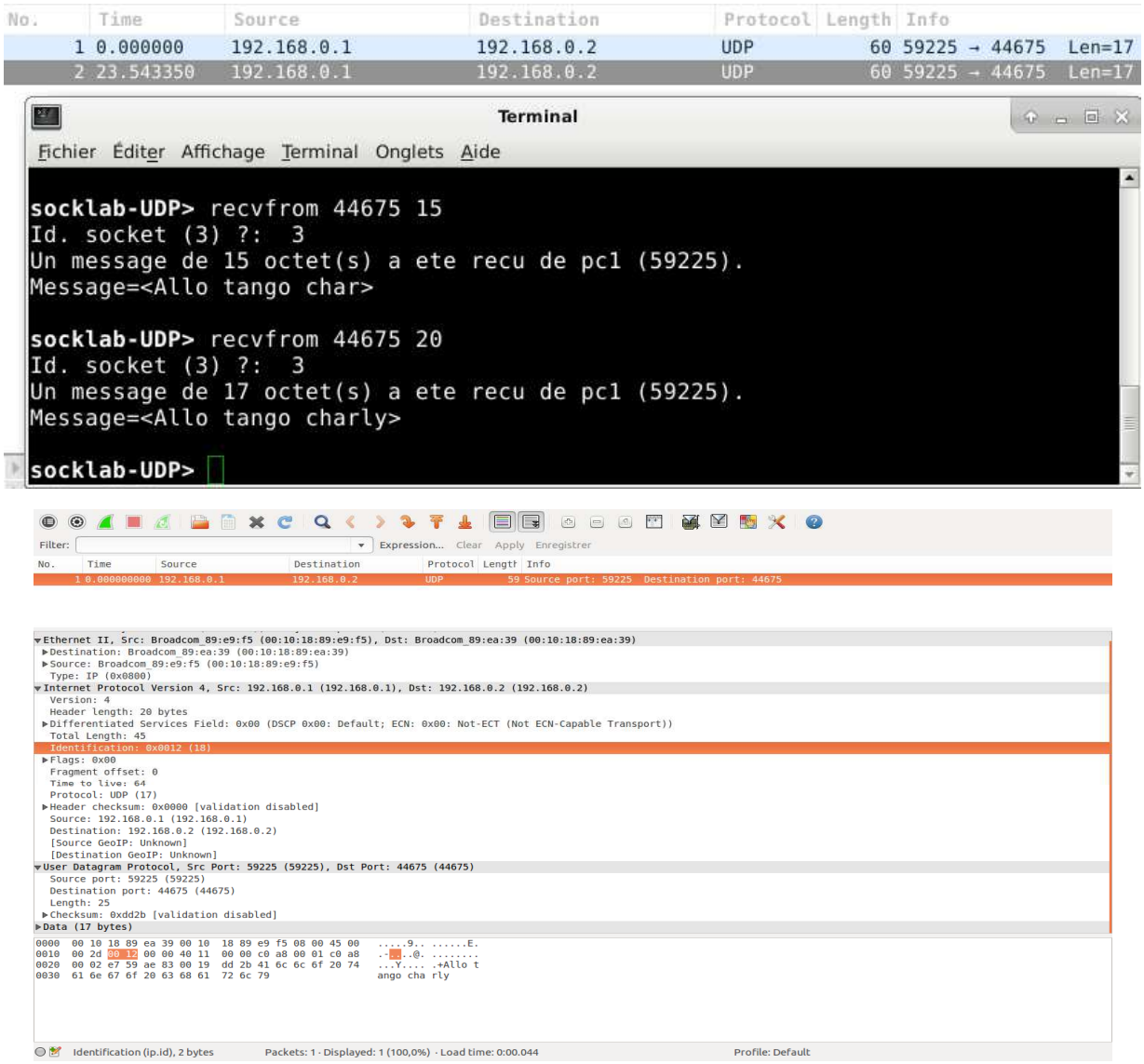
Adresse IP : 192.168.0.2 (PC2)

Port : 44675

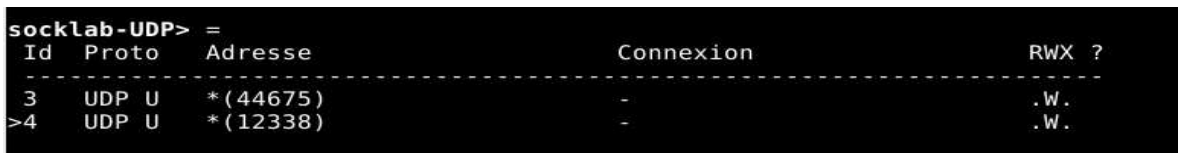
Id : 3 ;

Q2) Nous constatons qu'un seul paquet a été engendré et malgré qu'il soit entièrement reçu par le PC2, seule la partie correspondant à la taille du buffer autorisé par la machine de réception est affiché à l'écran.

Message envoyé du PC1 au PC2 : “Allo Tango Charly” (voir les captures du terminal et du Wireshark)



Q3) L’indicateur de socket permet de lister une association locale entre le port et l’adresse IP, indépendamment de la gestion purement locale.



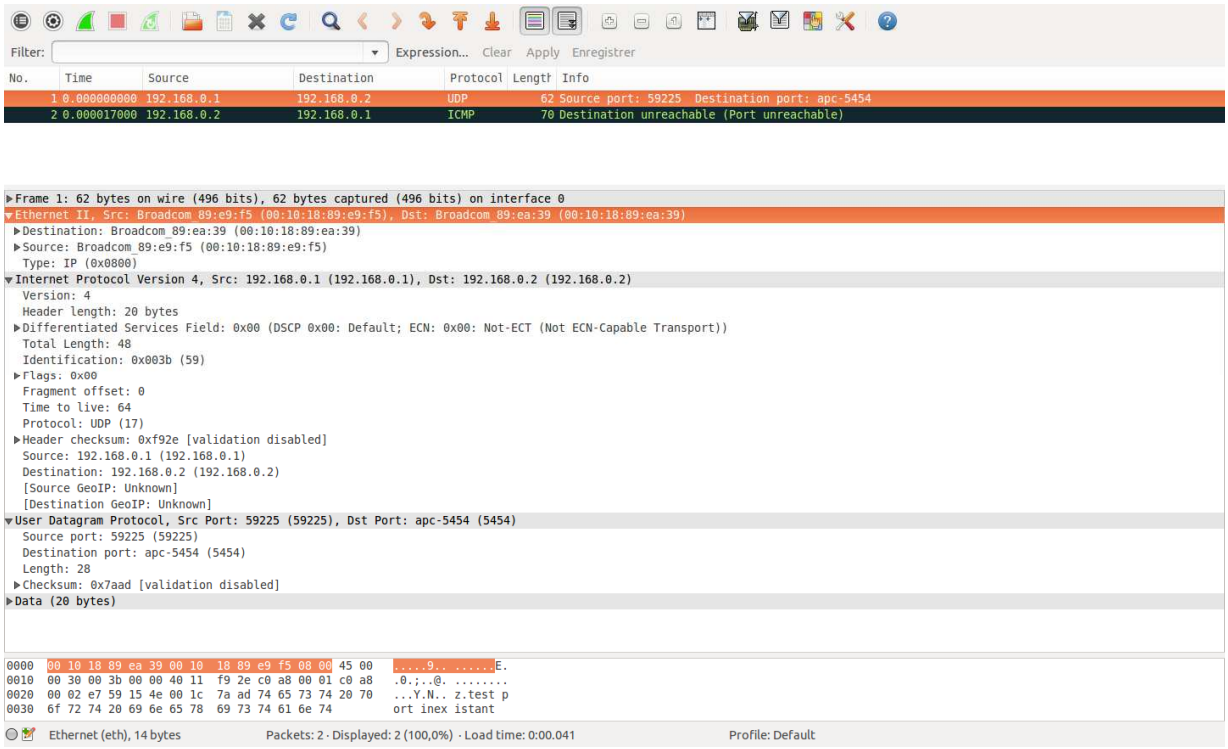
Q4) UDP donne les paramètres de services IP, mais dans lors des détails de fin ces informations ne sont jamais affichées.

Q5) Nous constatons qu'en demandant la réception avant l'émission des données la machine réceptrice se met en attente de la réception. Et lorsque c'est la machine émettrice qui envoie les données, elle le fait avec autant de paquets que nécessaire car un échange de données à l'aide d'UDP ne nécessite pas d'acquittement de réception des paquets. Chacune des machines reçoit le message qui lui est dédié proportionnellement à la taille de son buffer autorisé. La taille du paquet envoyé se doit être inférieure ou égal à la taille du buffer de réception, cependant le buffer de lecture ne peut afficher que le message proportionnel à sa taille, donc ce qu'une partie du message qui s'affiche lorsque le buffer de la réception est plus grand que le buffer de lecture.

Lorsque le message est envoyé vers un port et un id qui n'existe pas matériellement sur le réseau, le message se perd tout simplement. Cependant lorsque si le message est envoyé vers un port inexistant, sur une machine identifiée, du coup la machine créée un paquet avec un protocole ICMP de type **Destination Unreachable** et de code **Port** Unreachable, qu'il renvoie à la machine émettrice. Et dans la partie donnée de ce paquet ICMP, il y a les entêtes IP et UDP du paquet précédent.

NB : la commande "***options***" permet de connaitre la taille des buffers de réception.

Voir la capture suivante pour matérialiser le résumé ci-dessus :



CONFIGURATION DES MACHINES POUR LE PROTOCOLE TCP:

Machine 1 :

Adresse IP : 192.168.0.1 (PC1) ;

Mode passive,

Port : 33146;

Id : 3 ;

Machine 2 :

Adresse IP : 192.168.0.2 (PC2) ;

Mode : active

Id : 3,

Port : 45746

Q6) Le premier socket créé représente la machine en mode **passive**, c'est la machine à l'écoute et s'attend à recevoir des données (voir capture suivante).

```
socklab-TCP> accept
Id. socket (3) ?:  3
Un appel de pc2 (45746) a ete intercepte.
La connexion est etablie sous l'identificateur 4.

socklab-TCP> =
 Id  Proto   Adresse                  Connexion                RWX ?
 ------------------------------------------------------------------
 3   TCP     *(33146)                 -                        ...
>4   TCP     pc1(33146)               pc2(45746)               .W.
```

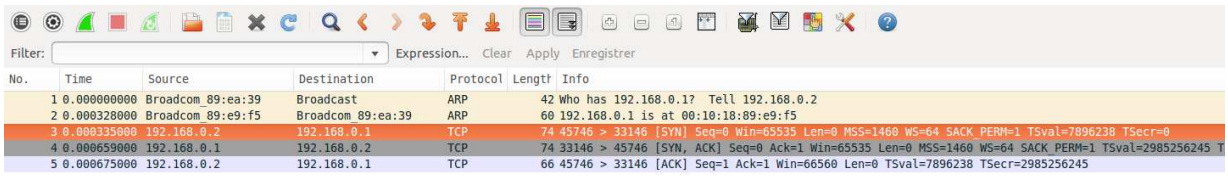
Tandis que le second socket représente la machine en mode **active**, c'est la machine émettrice (voir capture suivante).

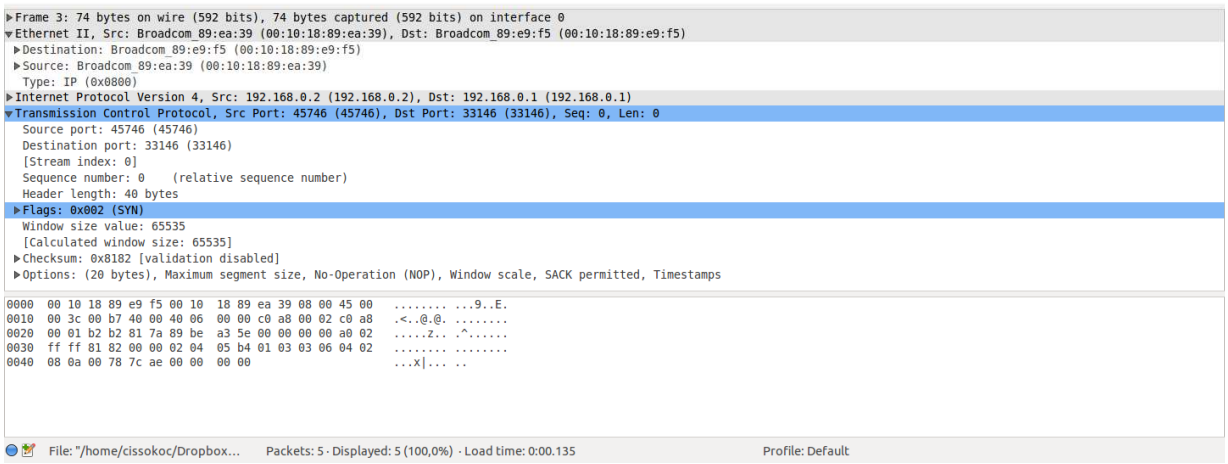
```
socklab-TCP> connect pc1 33146
Socket TCP creee: id=3, port=45746
Connexion etablie.

socklab-TCP> =
 Id  Proto   Adresse                  Connexion                RWX ?
 ------------------------------------------------------------------
>3   TCP     pc2(45746)               pc1(33146)               .W.
```

Q7) Nous distinguons trois étapes de cette connexion :

La première étape correspond à une demande de connexion de la machine en mode ***active*** (***le client***) vers la machine en mode ***passive***(***le serveur***) correspondant à la première TCP de notre capture ci-dessous. La seconde étape correspond à une réponse (acceptant ou refusant la connexion) de la machine en mode ***passive*** vers la machine en mode ***active***, donc un acquittement du serveur (deuxième ligne TCP de la capture ci-dessous). La dernière étape correspond à une sorte de confirmation de bonne réception, que la machine ***active*** émet vers la machine ***passive***, donc une confirmation de l'acquittement du serveur, envoyé du client vers le serveur(troisième ligne TCP de la capture ci-dessous.

Voir les trois (3) paquets TCP : ***SYN, SYN-ACK, ACK*** :





Q8) (**A revoir**)

Le flag SYN permet d'identifier le paquet comme étant une demande de connexion.

Le tableau ci-dessous décrit les échanges durant ces étapes.

NUMERO ET TYPES DE PAQUETS	**SEQUENCE NUMBER**	**ACKNOWLEDGMENT NUMBER**
Paquet 1, SYN	89 be a3 5e (relatif 0)	0
Paquet 2, SYN, ACK	70 7d 59 87 (relatif 0)	89 be a3 5f (relatif 1)
Paquet 3, ACK	89 be a3 5f (relatif 1)	70 7d 59 88(relatif 1)

Les numéros d'acquittement : ….

Numéros de séquence : ….

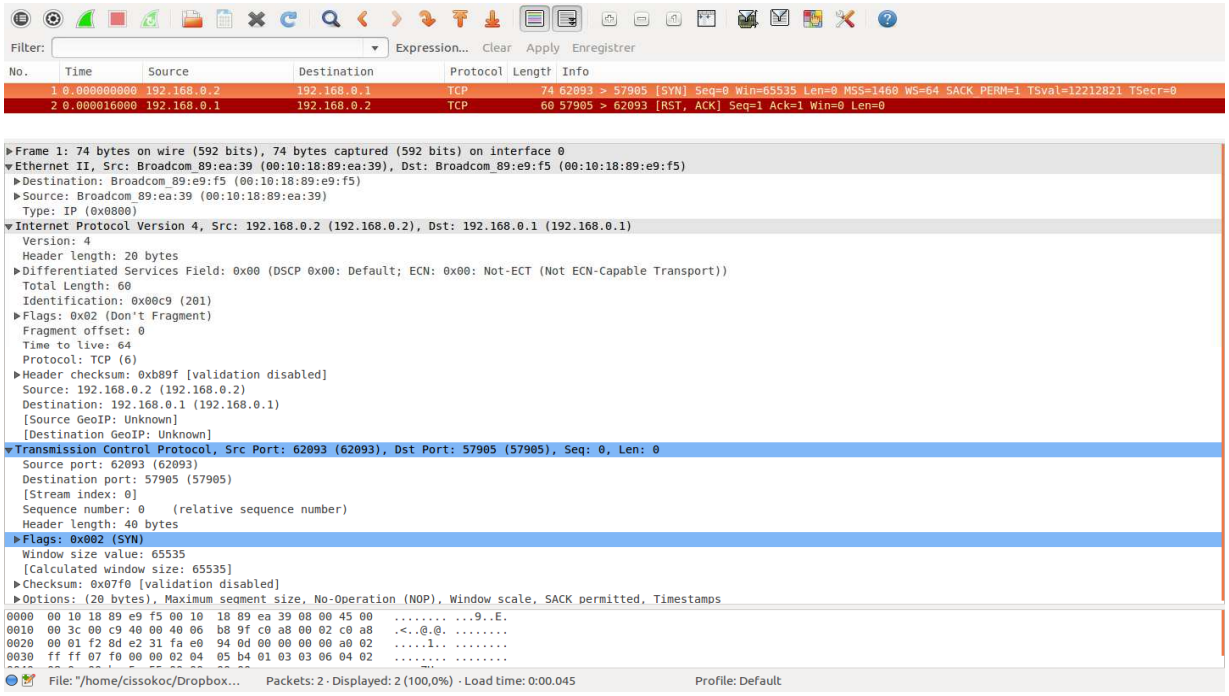
Q9) Que la commande "***accept***" soit réalisée avant ou après la commande "***connect***" nous constatons la création de la connexion entre les deux machines. A chaque nouvelle demande de connexion d'un client on a un nouveau numéro de port (socket) qui est créé systématiquement quand le serveur définit un port spécifique pour un client donné (voir la capture ci-dessous).

```
socklab-TCP> =
 Id   Proto    Adresse                    Connexion                  RWX ?
 ------------------------------------------------------------------------
 3    TCP      *(57906)                   -                          ...
>4    TCP      pc1(57906)                 pc2(49964)                 .W.

socklab-TCP> accept
Id. socket (4) ?:  3
Un appel de pc2 (35682) a ete intercepte.
La connexion est etablie sous l'identificateur 5.

socklab-TCP> =
 Id   Proto    Adresse                    Connexion                  RWX ?
 ------------------------------------------------------------------------
 3    TCP      *(57906)                   -                          ...
 4    TCP      pc1(57906)                 pc2(49964)                 .W.
>5    TCP      pc1(57906)                 pc2(35682)                 .W.
```

Q10) Une connexion est réellement identifiée par les **adresses** et **numéros de port** des machines connectées, d'où un minimum de quatre informations pour deux machines connectées entre elles.

Q11) L'*état de la connexion (**state**)* définit s'il existe une connexion entre deux sockets, chaque socket correspond à une machine distincte (***ESTABLISHED***). La machine jouant le rôle serveur a un socket en écoute seulement (***LISTEN***) ;

Q12) La machine émettrice crée un socket direct sans réfléchir pour demande de connexion et la machine réceptrice renvoie un flag "**RST, ACK"** à destination de celle émettrice qui supprime du coup son socket.



Q13) Actif pc1 port 55006, Passif pc2 port 45362
Données envoyées 5000 octets

Paquet (numero)	Source	Seq number	Ack number
1	Pc1	1 (0153 1C7B)	1 (ABDD 495E)
2	Pc1	1449 (0153 2223)	1 (ABDD 495E)
3	Pc2	1 (ABDD495E)	2897 (0153 27CB)
4	Pc1	2897 (0153 27CB)	1 (ABDD 495E)
5	Pc2	1 (ABDD 495E)	4345 (0153 2D73)
6	Pc1	4345 (0153 2D73)	1 (ABDD 495E)
7	Pc2	1 (ABDD 495E)	5001 (0153 3003)

Le champ ack number est utilisé par la machine émettrice pour identifier les paquets correspondant au même message. Sur le tableau ci-dessus l'ack number de pc1 est toujours 1 (ABDD 495E). Cet identifiant sur 2 octets est utilisé par la machine réceptrice (pc2) dans le champ seq number des paquets qu'elle envoie pour accuser la bonne réception. Le champ seq number est utilisé par la machine émettrice pour identifier la position du premier byte de données dans le message. La machine réceptrice met dans ack number le numéro des derniers bytes +1 qu'elle a reçu. (elle a tout reçu du premier byte au byte en « cours »)

Q14) Non un acquittement peut acquitter plusieurs paquets envoyés dans le tableau précédent on voit que le paquet 3 acquitte les deux paquets précédents. (image WS)

Q15) Le protocole TCP utilise un timer pour lors de l'envoie de donné. Quand ce timer sonne, un paquet est réémis. Dans le cas ou la machine réceptrice n'est pas à l'écoute, l'envoie d'un nouveau paquet se fait à chaque sonnerie de timer. Pour éviter de saturer le réseau le timer est double à chaque réémission d'un même paquet. (image WS).

A comparer avec la même manip en UDP

Q16) Voir cours
Q17) Voir cours
Q18) Voir cours
Q19) Voir cours
Q20) Voir cours
Q21) Voir cours
Q22) Vue simplifiée : le client envoie des paquets jusqu'à ce que le buffer de réception du receveur soit plein. Le receveur indique par un paquet zerowindow que son buffer de reception est plein. Une fois qu'un read a été effectué sur le receveur, alors le client envoie des données correspondant au maximum de la taille libérée dans le buffer.
Q23)
Q24)
Q25)

Q26)

Q27) Close du coté client en premier (q27 fermeture client)

Role du flag FIN : le paquet contenant le flag FIN indique que son émetteur a fermé sa connexion.

Remarque : si connexion pas accepté et fermeture serveur (q27 fermeture client puis serveur non accept) envoie paquet flag rst, ack.

Pc2 serveur fermé en premier

Paquet num	Source	Seq number	Ack number
1	Pc2	1 (149E 92E4)	1 (9C64 33A5)
2	Pc1	1 (9C64 33A5)	2 (149E 92E5)
3	Pc1	1 (9C64 33A5)	2 (149E 92E5)
4	Pc2	2 (149E 92E5)	2 (9C64 33A6)

Q28) Utiliser tableau pour cette réponse.
Q29) Remarque la procédure est la même quel que soit l'ordre de fermeture des connexions
Q30)
Q31)
Q32)
Q33)