Compte rendu tp4 réseaux première partie

Création du réseaux

On créer le réseau comme au tp2, c'est à dire deux machines vm1 et vm3 dans deux sous réseaux différents reliés par un routeur vm2.

Service echo

On installe le service echo sur la vm3 et on le test avec telnet depuis vm1.
Voici la capture de trames depuis vm2 :

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.16.2.131	172.16.2.163	TCP	74	50297→7 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1
2	0.000231000	172.16.2.163	172.16.2.131	TCP	74	7→50297 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460
3	0.000403000	172.16.2.131	172.16.2.163	TCP	66	50297→7 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=30250 TS
4	8.262364000	172.16.2.131	172.16.2.163	ECHO	81	Request
5	8.262599000	172.16.2.163	172.16.2.131	TCP	66	7→50297 [ACK] Seq=1 Ack=16 Win=29056 Len=0 TSval=229010
6	8.262625000	172.16.2.163	172.16.2.131	ECHO	81	Response
7	8.262738000	172.16.2.131	172.16.2.163	TCP	66	50297→7 [ACK] Seq=16 Ack=16 Win=29312 Len=0 TSval=32316

Client/serveur simples

Quelle est la différence entre terminer le client par `Ctrl-C` ou `Ctrl-D`?

Terminer le client par ctrl+d lui permet de prévenir le serveur de la fermeture de la connexion. En effet ctrl+d envoi le caractère de fin de fichier 'EOF' sur l'entré du client.
Alors que terminer le client par ctrl+c envoi le signal SIGINT au processus ce qui a pour effet de 'tuer' le processus.

Que passe-t-il si deux clients se connectent en même temps? Pourquoi?

Quand deux client souhaitent se connecter au serveur en même temps, le premier y parvient mais pas le second. En effet, pour le moment le serveur ne gère pas les connections simultanées car la fonction recv qui écoute sur le socket du premier client est bloquante.

Sur quel port écoute votre serveur? Comment cela est-il configurable?

Le serveur écoute sur le port donné en paramètre de la commande.

Vm3 utilise les ports suivants :

Proto	Recv-Q	Send-Q	Adresse locale	Adresse distante	Etat	PID/Program name
tcp	0	0	0.0.0.0:22	0.0.0.0:*	LISTEN	-
tcp	0	0	127.0.0.1:25	0.0.0.0:*	LISTEN	-
tcp	0	0	0.0.0.0:35296	0.0.0.0:*	LISTEN	-
tcp	0	0	0.0.0.0:7	0.0.0.0:*	LISTEN	-
tcp	0	0	0.0.0.0:111	0.0.0.0:*	LISTEN	-
tcp	0	0	0.0.0.0:1234	0.0.0.0:*	LISTEN	2565/echoserveur.exe
tcp6	0	0	:::22	:::*	LISTEN	-
tcp6	0	0	::1:25	:::*	LISTEN	-
tcp6	0	0	:::38339	:::*	LISTEN	-
tcp6	0	0	:::111	:::*	LISTEN	-
udp	0	0	0.0.0.0:46973	0.0.0.0:*		-
udp	0	0	0.0.0.0:1003	0.0.0.0:*		-
udp	0	0	127.0.0.1:1013	0.0.0.0:*		-
udp	0	0	0.0.0.0:62016	0.0.0.0:*		-
udp	0	0	0.0.0.0:68	0.0.0.0:*		-
udp	0	0	0.0.0.0:111	0.0.0.0:*		-
udp6	0	0	:::60817	:::*		-
udp6	0	0	:::1003	:::*		-
udp6	0	0	:::36355	:::*		-
udp6	0	0	:::111	:::*		-

Pouvez-vous lancer deux serveurs en même temps?

On peut lancer deux serveur sur la même machine à condition de spécifier des ports différents en paramètre.

Le serveur peut-il faire la différence avec `echoclient`? Y a-t-il une différence avec `echoclient`?

Le serveur ne peut pas faire la différences entre une trame envoyé par echoclient (c client) et EchoClient (java client) car elles sont identiques dans leur structure (paquet IPV4).
Il n'y a pas de différence entre le client telnet et le client echoclient dans le sens ou ils envoient les même paquet IPV4 et qu'il attendent de recevoir un message du serveur après chaque envoi et l'affiche.

En quoi cette partie illustre-t-elle le paradigme client/serveur?

Le serveur ne fait que répondre aux requêtes du client, ici renvoyer le message reçu par un client. Ce qui fait que le client se compote toujours de la même façon, il ne fait qu'envoyer des messages et attend que le serveur lui renvoi avant de l'afficher. Le serveur est un service et les clients lui envois des données à traiter.

Api socket et tcp/ip

Voici une capture de trame depuis vm3 :

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.16.2.131	172.16.2.163	TCP	74	46361→1234 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=393472 TSecr=0 WS=128
2	0.000384000	172.16.2.163	172.16.2.131	TCP	74	1234→46361 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=373566 TSecr=393472
3	0.000605000	172.16.2.131	172.16.2.163	TCP	66	46361→1234 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=393473 TSecr=373566
4	0.002193000	172.16.2.163	172.16.2.131	TCP	102	1234→46361 [PSH, ACK] Seq=1 Ack=1 Win=29056 Len=36 TSval=373567 TSecr=393473
5	0.002435000	172.16.2.131	172.16.2.163	TCP	66	46361→1234 [ACK] Seq=1 Ack=37 Win=29312 Len=0 TSval=393473 TSecr=373567
6	3.760033000	172.16.2.131	172.16.2.163	TCP	72	46361→1234 [PSH, ACK] Seq=1 Ack=37 Win=29312 Len=6 TSval=394413 TSecr=373567
7	3.760469000	172.16.2.163	172.16.2.131	TCP	66	1234→46361 [ACK] Seq=37 Ack=7 Win=29056 Len=0 TSval=374506 TSecr=394413
8	3.760562000	172.16.2.163	172.16.2.131	TCP	74	1234→46361 [PSH, ACK] Seq=37 Ack=7 Win=29056 Len=8 TSval=374506 TSecr=394413
9	3.761019000	172.16.2.131	172.16.2.163	TCP	66	46361→1234 [ACK] Seq=7 Ack=45 Win=29312 Len=0 TSval=394413 TSecr=374506
10	4.549526000	172.16.2.131	172.16.2.163	TCP	66	46361→1234 [FIN, ACK] Seq=7 Ack=45 Win=29312 Len=0 TSval=394610 TSecr=374506
11	4.550009000	172.16.2.163	172.16.2.131	TCP	80	1234→46361 [PSH, ACK] Seq=45 Ack=8 Win=29056 Len=14 TSval=374704 TSecr=394610
12	4.550437000	172.16.2.131	172.16.2.163	TCP	66	46361→1234 [ACK] Seq=8 Ack=59 Win=29312 Len=0 TSval=394610 TSecr=374704

Voici ce qui est échangé :

→ Les 3 premières trames sont la « poignée de main » du client pour initialiser la connexion(connexion du client, acceptation du serveur, accusé de réception du client).

→ Les trames 4 et 5 sont l'envoi du message de bienvenue du serveur au nouveau client suivit de l'accusé de réception du client.

→ Les trames 6 et 7 sont l'envoi d'un message du client au serveur suivit de l'accusé de réception du serveur.

→ Les trames 8 et 9 sont l'envoi du même message du serveur au premier client suivis de l'accusé de réception du client.

→ Les 3 dernières trames correspondent à l'envoi du client d'un message pour prévenir le serveur qu'il se déconnecte. Le serveur lui répond qu'il a bien reçu l'information et le client envoi un accusé de réception.