STI 3ème année – Réseau

TD 2: Client / Server UDP et TCP

J. Briffaut

L'objectif de ce TD est d'écrire :
— Un client / serveur UDP simple
— Un client / serveur TCP simple monoclient

1 UDP

Commençons par un PING-PONG udp. Un client enverra en UDP un message PING, que le serveur affichera, puis enverra un PONG au client, que celui affichera, puis le client enverra PING .... jusqu'a un ctrl+c pour stopper les programmes

1.1 Server

Vous allez créer un serveur UDP écoutant sur le port 1234 et l'adresse localhost. Le serveur devra attendre un message, l'afficher, puis envoyer PONG, et se remettre en attente d'un message.

Exercice 1 Créez un fichier **udpserver.c**

Exercice 2 Ajouter une fonction **stop** habituelle pour gérer correctement les erreurs d'appels système

Exercice 3 Créez une socket (**socket()**) de type **UDP** IPv4

Exercice 4 Créez une structure d'adresse (**sockadd_in**) pour l'adresse **127.0.0.1** et le port **1234**

Exercice 5 Associez votre socket a cette adresse (**bind()**)

Exercice 6 Ajouter une boucle infini qui
- attend la réception d'un message (**recvfrom**)
- affiche le message reçu
- affiche l'ip et le port de l'émetteur
- attend 1 seconde
- envoi le message "PONG" (**sendto**)

Exercice 7 Tester votre serveur avec **echo -n coucou | nc -4u -w0 localhost 1234**

1.2 Client

Vous allez créer un client UDP envoyant des messages sur le port 1234 et l'adresse localhost. Le client enverra PING, puis devra attendre un message, l'afficher, et se remettre a envoyer PING.

Exercice 8 Créez un fichier **udpclient.c**

Exercice 9 Ajouter une fonction **stop** habituelle pour gérer correctement les erreurs d'appels système

Exercice 10 Créez une socket (**socket()**) de type **UDP** IPv4

Exercice 11 Créez une structure d'adresse (**sockadd_in**) pour l'adresse correspondant au serveur **127.0.0.1** et le port **1234**

Exercice 12 Ajouter une boucle infini qui
- envoi le message "PING" au serveur (**sendto**)
- attend la réception d'un message (**recvfrom**)
- affiche le message reçu

Exercice 13 Tester votre client avec votre serveur (vous pouvez aussi vérifier avec Wireshark)

2 TCP

Vous aller créer un serveur ECHO, tout message envoyé au serveur serra re-envoyé au client.

2.1 Server

Vous allez créer un serveur TCP écoutant sur le port 1234 et l'adresse localhost. Le serveur devra attendre un message, l'afficher, puis le renvoyer se remettre en attente d'un message.

Exercice 14 Créez un fichier **tcpserver.c**

Exercice 15 Ajouter une fonction **stop** habituelle pour gérer correctement les erreurs d'appels système

Exercice 16 Créez une socket (**socket()**) de type **TCP** IPv4

Exercice 17 Créez une structure d'adresse (**sockadd_in**) pour l'adresse **127.0.0.1** et le port **1234**

Exercice 18 Associez votre socket a cette adresse (**bind()**)

Exercice 19 Ecoutez au maximum 5 clients (**listen()**)

Exercice 20 Acceptez un client (**accept()**)

Exercice 21 Ajouter une boucle de 1000 itérations qui
- attend la réception d'un message (**recv**)
- affiche le message reçu
- renvoi le message (**send**)

Exercice 22 Tester votre serveur avec **echo -n coucou | nc -4 -w0 localhost 1234**

Exercice 23 Que ce passe t'il ? Comment corriger ce problème ?

2.2 Client

Vous allez créer un client TCP envoyant des messages sur le port 1234 et l'adresse localhost. Le client enverra ECHO, puis devra attendre un message, l'afficher, et se remettre a envoyer ECHO, ceci 1000 fois.

Exercice 24 Créez un fichier **tcpclient.c**

Exercice 25 Ajouter une fonction **stop** habituelle pour gérer correctement les erreurs d'appels système

Exercice 26 Créez une socket (**socket()**) de type **TCP** IPv4

Exercice 27 Créez une structure d'adresse (**sockadd_in**) pour l'adresse correspondant au serveur **127.0.0.1** et le port **1234**

Exercice 28 Ajouter une boucle de 1000 itérations qui

- envoi le message "ECHO" au serveur (**send**)
- attend la réception d'un message (**recv**)
- affiche le message reçu

Exercice 29 Tester votre client avec votre serveur (vous pouvez aussi vérifier avec Wireshark)