M2 FEADéP – Systèmes et réseaux
TP6 Sockets III

Nicolas Chappe
École Normale Supérieure de Lyon

1 Complément sur les sockets

Nous avons jusqu'à maintenant travaillé avec des sockets TCP, qui nécessitent l'établissement d'une connexion entre client et serveur. Ce n'est pas le cas pour la communication UDP : il est possible d'envoyer des paquets à un serveur sans avoir préalablement établi de connexion. Cela simplifie son utilisation par rapport à TCP, mais vient avec d'autres désavantages. [1]

Pour envoyer un paquet UDP, `ssize_t sendto(int sockfd, const void *buf, size_t len, int flags, const struct sockaddr *dest_addr, socklen_t addrlen)` a la même signature que `send`, plus les arguments `dest_addr` et `addrlen` qui permettent de spécifier l'adresse de destination du paquet.

De la même manière, `ssize_t recvfrom(int sockfd, void *buf, size_t len, int flags, struct sockaddr *src_addr, socklen_t *addrlen)` est analogue à `recv` mais place dans `src_addr` l'adresse d'origine du paquet reçu. Le fonctionnement de `addrlen` est un peu plus complexe : c'est un pointeur vers un entier contenant l'espace disponible dans `src_addr` (parce que toutes les adresses ne font pas la même taille, par exemple IPv4 vs IPv6).

Une autre subtilité de `recvfrom` est que si `buf` n'est pas assez grand pour contenir le paquet reçu, la fin du paquet sera perdue.

2 Ping pong

Exercice 1

Le fichier `pong.c` fourni implante un serveur qui, lorsqu'il reçoit un paquet UDP contenant exactement la chaîne `"ping"` sur le port 8325, répond en envoyant un paquet contenant la chaîne `"pong"` à l'émetteur [2]. À vous d'implanter un client qui demande d'entrer l'adresse du serveur, lui envoie un tel paquet et vérifie qu'il reçoit bien la réponse attendue.

1. Pour rappel, le protocole UDP présente une moindre fiabilité (les paquets peuvent être reçus dans le désordre, plusieurs fois, ou simplement perdus).
2. La vraie commande `ping` fonctionne un peu différemment, elle n'utilise pas le protocole UDP mais le protocole ICMP.

Pourquoi ne pas l'écrire en Python pour changer ? Vous aurez besoin de la classe `socket` du module socket et des méthodes `sendto` et `recvfrom`, le code tient en une dizaine de lignes.

À noter qu'en Python, les fonctions de la famille recv/send fonctionnent non pas avec des chaînes de caractères de type `str` mais avec des objets de type bytes plus bas niveau qui représentent une suite d'octets quelconque. Pour par exemple créer un `bytes` de 4 octets contenant `"ping"`, on peut utiliser la syntaxe `b'ping'`.