Sujet de TP 10 Réseaux - Sockets L3 Informatique/UFR ST/UFC

Programmation réseau avec sockets en C sous Linux avec IPv6 (5)

V. Felea & A. Hugeat & E. Merlet

Deux autres familles de protocoles implémentées sous Linux existent, correspondant d'une part à la communication locale (AF_UNIX), et d'autre part à la communication distante pour le protocole internet IPv6.

Ci-après quelques indications pour l'utilisation des sockets dans la communication distante basée sur le protocole IPv6.

Adresse IPv6 Le type d'adresse d'une socket de communication distante en IPv6 est défini dans la bibliothèque `netinet/in.h` (ou dans la page du manuel `man ipv6`) :

```
struct sockaddr_in6 {
  sa_family_t     sin6_family;   /* AF_INET6 */
  in_port_t       sin6_port;     /* port number */
  uint32_t        sin6_flowinfo; /* IPv6 flow information */
  struct in6_addr sin6_addr;     /* IPv6 address */
  uint32_t        sin6_scope_id; /* scope ID */
};

struct in6_addr {
  unsigned char   s6_addr[16];   /* IPv6 address */
};
```

où

- `sin6_family` est toujours égal à `AF_INET6`,
- `sin6_port` est le numéro de port (ordre d'octets réseau),
- `sin6_addr` est l'adresse IPv6 de la socket. Le serveur peut utiliser la variable globale `in6addr_any` pour initialiser l'adresse de la socket. Le client peut l'initialiser avec la fonction `inet_pton`.
- `sin6_scope_id` est le numéro d'interface sortante du client. Peut être initialisée grâce à la fonction `if_nametoindex` (bibliothèque <`net/if.h`>).

Les autres champs peuvent être initialisés à 0.

Q1 **Binaires IPv6 client/serveur** Télécharger les binaires IPv6 (depuis le cours Réseaux Moodle), correspondant à un client et un serveur de l'exercice 1 du TP6, en utilisant l'adressage IPv6. Lancer les binaires, en utilisant premièrement l'interface réseau locale et deuxièmement, l'interface Ethernet.

Q2 **Programme client en mode connecté** Développer le client. Le tester avec le binaire serveur IPv6 fourni. Le test utilisera aussi dans un premier temps l'interface réseau locale et dans un deuxième temps, l'interface Ethernet.

Q3 **Programme serveur en mode connecté** Développer le serveur et le tester avec le client développé par vos soins.

Conversions (rappel) La fonction `inet_aton` convertit des adresses pour IPv4 (voir cours), à partir des notations diverses (décimale à point, hexadécimale, octale). La fonction `inet_pton` traite aussi des adresses IPv6, les adresses IPv4 étant uniquement en notation décimale à point. Pour rendre cette conversion plus générique, indépendamment du type d'adresse (IPv4 ou IPv6), et être capable de résoudre des noms de domaine, utiliser la fonction `getaddrinfo`(3) (voir exemple dans le cours).

Q4 **Communication IPv4-IPv6** Dans cette partie, nous sommes intéressés à répondre à la question "Est-il possible de faire communiquer une application IPv4 avec une application IPv6 ?" Faire deux tests croisés pour la même application (Exercice 1 du TP6) :

- le client IPv4 avec le serveur IPv6,
- le client IPv6 avec le serveur IPv4.

Conclure.