**David TREMBLAY 1748125**

**Mohamed Seghaier LAMOUCHI 1765912**

# Laboratoire 3

**INF3405**

**Réseaux informatiques**

**Groupe 01**

**Département de génie informatique et génie logiciel**

**École polytechnique de Montréal**

**23 mai 2017**

## Nom du poste : L4708-18

## Question A

**1)** Le poste contient **9** adresses IP

```
getaddrinfo a reussi!
Adresse # 1
        Flags: 0x0
        Famille: AF_INET6 (IPv6)
        Adresse IPv6: fe80::e8ff:3245:e20c:d1b7
        Taille de cette adresse: 28 octets
        Nom canonique: (null)
Adresse # 2
        Flags: 0x0
        Famille: AF_INET6 (IPv6)
        Adresse IPv6: fe80::85ff:18f1:4b3e:399
        Taille de cette adresse: 28 octets
        Nom canonique: (null)
Adresse # 3
        Flags: 0x0
        Famille: AF_INET6 (IPv6)
        Adresse IPv6: fe80::d5ee:6cfe:31c7:a974
        Taille de cette adresse: 28 octets
        Nom canonique: (null)
Adresse # 4
        Flags: 0x0
        Famille: AF_INET6 (IPv6)
        Adresse IPv6: fe80::bd23:1415:4849:7f28
        Taille de cette adresse: 28 octets
        Nom canonique: (null)
Adresse # 5
        Flags: 0x0
        Famille: AF_INET (IPv4)
        Adresse IPv4: 132.207.29.118
        Taille de cette adresse: 16 octets
        Nom canonique: (null)
Adresse # 6
        Flags: 0x0
        Famille: AF_INET (IPv4)
        Adresse IPv4: 192.168.44.60
        Taille de cette adresse: 16 octets
        Nom canonique: (null)
Adresse # 7
        Flags: 0x0
        Famille: AF_INET (IPv4)
        Adresse IPv4: 192.168.233.1
        Taille de cette adresse: 16 octets
        Nom canonique: (null)
Adresse # 8
        Flags: 0x0
        Famille: AF_INET (IPv4)
        Adresse IPv4: 192.168.142.1
        Taille de cette adresse: 16 octets
        Nom canonique: (null)
Adresse # 9
        Flags: 0x0
        Famille: AF_INET6 (IPv6)
        Adresse IPv6: 2002:84cf:1d76::84cf:1d76
        Taille de cette adresse: 28 octets
        Nom canonique: (null)
Appuyer une touche pour finir...
```

**2)** On a plusieurs adresses IP parce **qu'un nœud peut contenir plusieurs interfaces**

**3)** **Classe A :**

**Plage : 0.0.0.0 à 127.255.255.255**
**Masque : 255.0.0.0**

**Classe B :**

**Plage : 128.0.0.0 à 191.255.255.255**
**Masque : 255.255.0.0**

**Classe C :**

**Plage : 192.0.0.0 à 223.255.255.255**
**Masque : 255.255.255.0**

**Classe D :**

**Plage : 244.0.0.0 à 239.255.255.255**
**Masque : Non défini**

**Classe E :**

**Plage : 240.0.0.0 à 255.255.255.255**
**Masque : Non défini**

**4**) L'adresse IPv4 de l'interface active sur le réseau de l'école est une **classe B.**

**5)**

**a.** **Adresse IP : 10000100.11001111.00011101.01110110**
**Masque : 11111111.1111111.11111111.0000000**

**b.** **Adresse du réseau : 10000100.11001111.00011101.0000000**
**En décimal : 132.207.29.0**

```
X:\>ipconfig

Configuration IP de Windows


Carte Ethernet Ethernet 3 :

   Suffixe DNS propre à la connexion. . . : lerb.polymtl.ca
   Adresse IPv6 de liaison locale. . . . .: fe80::e8ff:3245:e20c:d1b7%9
   Adresse IPv4. . . . . . . . . . . . . .: 132.207.29.118
   Masque de sous-réseau. . . . . . . . . : 255.255.255.0
   Passerelle par défaut. . . . . . . . . : 132.207.29.1
```

**6)**

```
getaddrinfo a reussi!
Adresse # 1
        Flags: 0x0
        Famille: AF_INET (IPv4)
        Adresse IPv4: 132.207.29.103
        Taille de cette adresse: 16 octets
        Nom canonique: (null)
Appuyer une touche pour finir...
```

**7)** Le poste L4708-03 est sur le même sous-réseau que notre poste. L'adresse IP du poste L4708-03 commence avec les trois octets 132.207.29 et le masque du sous-réseau est 255.255.255.0. De plus, l'adresse de notre poste commence aussi avec les trois octets 132.207.29. On en conclut que les deux sont sur le même sous-réseau.

**8)** Le nom de la fonction qui permet de retrouver les adresses des interfaces associées à un nom d'hôte est **getaddrinfo().** Elle retourne, entre autres, **l'adresse et sa famille.**

## Question B

**9)** Le port utilisé pour communiquer avec le serveur est **5000**

**10)** La famille d'adresses utilisée pour le socket est **AF_INET**

**11)** Le type de socket crée est SOCK_STREAM. C'est un type utilisé par le protocole TCP. En d'autres termes, c'est un protocole basé sur la connexion contrairement au protocole UDP basé sur les datagrammes.

```
Adresse trouvee pour le serveur 132.207.29.118 : 132.207.29.118

Tentative de connexion au serveur 132.207.29.118 avec le port 5000

Connecte au serveur 132.207.29.118:5000

Saisir un mot de 7 lettres pour envoyer au serveur: network
Nombre d'octets envoyes : 7
Nombre d'octets recus: 7
Le mot recu est NeTwOrK
Appuyez une touche pour finir
```

Figure 1: Résultat obtenu

**12)** La fonction utilisée pour envoyer un mot au serveur est **send()**. Elle prend en paramètres le socket utilisé, le mot à envoyer, la longueur de celui-ci ainsi que le flag.

**13)** Le nom de la fonction qui permet de recevoir l'information est **recv()**

**14)** Dans la réponse du serveur, les lettres d'index pair sont mises en majuscules. (network → NeTwOrK)

**15)** Le client n'arrive pas à se connecter au serveur sur le port 5050 parce qu'il utilise le port 5050 mais le serveur écoute sur le port 5000.

```
Adresse trouvee pour le serveur 132.207.29.118 : 132.207.29.118

Tentative de connexion au serveur 132.207.29.118 avec le port 5050

Impossible de se connecter au serveur 132.207.29.118 sur le port 5050

Appuyez une touche pour finir
```

**16)** Le port utilisé par le client pour envoyer doit être le même que le port utilisé par le serveur pour recevoir afin d'établir une connexion.

```
Adresse trouvee pour le serveur 132.207.29.118 : 132.207.29.118

Tentative de connexion au serveur 132.207.29.118 avec le port 5050

Connecte au serveur 132.207.29.118:5050

Saisir un mot de 7 lettres pour envoyer au serveur: network
Nombre d'octets envoyes : 7
Nombre d'octets recus: 7
Le mot recu est NeTwOrK
Appuyez une touche pour finir
```

**17)**

Nous avons ajouté un tableau de caractères nommé IP qui va contenir l'adresse entrée par l'utilisateur avec la fonction gets_s. Nous avons aussi une chaîne de caractères nommée adresseEcole pour comparer les 13 premiers caractères de l'adresse entrée pour s'assurer qu'ils sont valides. SI ce n'est pas le cas, nous avons un message d'erreur et le programme s'arrête. Finalement, nous comparons les 2 derniers caractères de l'adresse IP qui doivent être entre 1 et 29. SI ce n'est pas le cas, un autre message s'affiche et le programme s'arrête.

```
char IP[15];
char derniers[2];
char adresseEcole[13] = "132.207.29.1";
printf("Entrez l'adresse IP\n");
gets_s(IP);
derniers[0] = IP[13];
derniers[1] = IP[14];
int nbr = atoi(derniers);
for (int i = 0; i < 13; i++)
{
    if (strcmp(IP[i]+"",adresseEcole[i]+"")) {
        printf("Mauvaise adresse\n");
        return -1;
}
}
if (nbr < 1 || nbr > 29) {
    printf("Mauvaise adresse2\n");
    return -1;
```

**18)** Le mécanisme utilisé pour traiter plusieurs connexions simultanément est le multithreading. À chaque connexion, le serveur crée un nouveau thread.

**19**) La fonction utilisée est **CreateThread()**

**20)**

| **Protocole** | **Port(s) par défaut** |
|---|---|
| HTTP | 80 |
| HTTPS | 443 |
| SSH | 22 |
| FTP | 20 |