Faculté des Sciences & Techniques — Université de Limoges

*Master 1ère année*

*Protocoles & Prog. Réseau*

TP n°2

*UDP & Communication multicast*

## Envoi de paquet UDP en multicast

La notion d'envoi « *multicast* » repose sur l'utilisation d'adresse IP associée à des **groupes**.
Les adresses de groupe sont des adresses de *classe D*, munies du préfixe binaire `1110`, ce qui donne des adresses comprises entre `224.0.0.0` et `239.255.255.255`.
Lorsqu'un datagramme IP est envoyé vers une adresse multicast, l'adresse Ethernet associée est de type `01:00:05E:xx:yy:zz`, où `xx:yy:zz` contiennent les 23 derniers bits de l'@IP du groupe.
*De par la nature multicast de l'envoi, il n'est pas possible d'utiliser le protocole TCP qui est un protocol unicast (c-à-d un vers un).*

Au niveau de la programmation *Socket*, on utilisera le protocole datagramme UDP et on associera cette adresse de groupe à un numéro de port spécifique.

Pour la réception, le système d'exploitation doit être informé de devoir récupérer les datagrammes UDP à destination du groupe spécifié :

```
1 gestion_mcast = struct.pack("4sl", socket.inet_aton("224.0.0.127"),
2                  socket.INADDR_ANY)
3 ma_socket.setsockopt(socket.IPPROTO_IP, socket.IP_ADD_MEMBERSHIP,
4                  gestion_mcast)
```

**Attention :** ⟹ c'est un « L » minuscule et pas le chiffre « 1 ».

Sous Python 3, vous pouvez aussi utiliser la procédure suivante pour `gestion_multicast` :

```
gestion_mcast = bytes([int(x) for x in b'224.0.0.127'.split(b'.')]+[0]*4)
```

*Ainsi, vous n'avez pas besoin du module `struct`.*

Dans les paramètres de l'instruction `bind`, il est possible de restreindre les réceptions aux datagrammes UDP envoyés **seulement** à un groupe donné :

```
ma_socket.bind(("224.0.0.251", 7182)) # où 7182 est le port choisi
```

*Les datagrammes UDP envoyés à destination de l'adresse IP et du port 7182 du destinataire ne seront pas reçus.*

## Conception d'un protocole de communication de groupe

Le travail va consister à concevoir un protocole très simple de communication de type *Chat* sur UDP en envoyant chaque message tapé par un utilisateur vers tous les autres utilisateurs en utilisant une adresse de groupe.

- ⋆ chaque message sera constitué d'une ligne construite de la manière suivante "`pseudo_emetteur:message`", exemple : `oisillon:Salut les poteaux !`
- ⋆ l'adresse IP de groupe est `224.0.0.127` ;
- ⋆ le numéro de port d'envoi et d'écoute est le `7182`.

### Réception de ses propres messages envoyés en multicast

Suivant son système d'exploitation et sa configuration, une machine qui fait partie d'un groupe peut ne pas recevoir les messages qu'elle transmets à destination de ce groupe.
Si on veut l'autoriser, il faut activer l'option de « boucle » :

```
ma_socket.setsockopt(socket.IPPROTO_IP, socket.IP_MULTICAST_LOOP, 1)
```

**1 –** Écrire un programme Python réalisant le protocole proposé. Il doit être capable de traiter les envois, comme les réceptions simultanément (utilisation de l'appel système `fork`).

**2 –** Écrire un programme Python réalisant le protocole étendu suivant :

- Pouvoir permettre de **cibler** les interlocuteurs auxquels on écrit.
- Il est nécessaire de :
  - **obtenir** de l'utilisateur le **pseudo** qu'il veut utiliser ;
  - **diffuser** à l'ensemble du groupe régulièrement le **pseudo** de l'utilisateur (par ex : toutes les 10s).
    *Il faut définir un nouveau message de la forme* `":pseudo"`
  - **découvrir la liste des pseudos** des utilisateurs du réseau local en écoutant les messages d'annonce de pseudo et afficher cette liste régulièrement à l'utilisateur ;
  - pour un message à envoyer par un utilisateur : **choisir entre le diffuser à tout le monde** ou bien le **transmettre uniquement à la machine** où l'utilisateur désigné par son pseudo est connecté.

    Le message est maintenant construit de la manière suivante :

    `pseudo_emetteur:pseudo_destination:message`

    exemple : `aigle_royal:oisillon:Salut, tu vas ?`

    *On pourra utiliser un message de la forme* `pseudo_emetteur:*:message` *pour envoyer le message à tout le monde.*

Par rapport à la version précédente, vous avez deux possibilités :

- envoyer les messages **toujours** à tout le monde (adresse IP groupe, port), et **seul le destinataire** du message affiche le message ;
- envoyer en « **unicast** » le message **uniquement** vers la machine de l'utilisateur.
  *Lorsqu'un pseudo est diffusé sur le réseau, chaque hôte peut noter l'@IP source du message, et associer ce pseudo à cette @IP.*

### Compléments : Portée et gestion de l'envoi multicast

Cet envoi de datagramme a une portée limitée à celle du réseau de la même structure administrative :

- il peut traverser un routeur si ce routeur est configuré pour l'autoriser ;
  *Pour limiter le nombre de routeurs traversables, on peut choisir le TTL du datagramme.*
- il peut joindre toutes machines connectées sur un « concentrateur filtrant » ou switch.
  *Pour informer le switch (ou le routeur), qu'une machine joint un groupe ou le quitte, des messages suivant le protocole IGMP, « Internet Group Management Protocol », sont transmis automatiquement par la machine.*

Exemple de trace d'un datagramme IP envoyé vers une adresse de groupe :

```
0000   01 00 5e 00 00 fb 00 17 f2 09 b0 56 08 00 45 00  ..^........V..E.
0010   00 20 f6 e0 00 00 01 11 00 00 c0 a8 01 32 e0 00  . ...........2..
0020   00 fb c6 18 14 ea 00 0c a2 f3 79 6f 20 3f        ..........yo ?
```

Sur cette trace, l'adresse Ethernet destination est `01:00:5e:00:00:fb` pour une adresse IP destination `224.0.0.251` ou en hexadécimal : `[e0 00 00 fb]`.
L'adresse source du paquet UDP est ici `192.168.1.50` ou en hexadécimal `[c0 a8 01 32]`.

*L'utilisation d'adresse multicast permet de faire de la découverte de voisinage : il n'est pas nécessaire de connaître l'adresse IP d'un destinataire pour le joindre et pour qu'il puisse répondre, ce qui permet de découvrir les autres hôtes connectés au même réseau local.*