# Travaux pratiques n° 4

## Communication de groupe

L'objectif du TP consiste à développer un logiciel offrant le service de messagerie instantanée sur un réseau local. Les codes associés seront écrits dans le langage Python

### Exercice 4.1 — Serveur de discussion en multicast IP

Nous allons reprendre l'idée de l'application de chat du premier TP. Cette application fonctionne toujours de manière interactive mais nous allons remplacer le serveur par une communication multicast.

Un utilisateur du logiciel peut rejoindre un salon de discussion, c.à.d un espace dans lequel l'ensemble des participants reçoit les mêmes messages. Nous supposons l'existence d'un salon unique. Par conséquent, un utilisateur ne peut participer qu'à un salon. Dans cette version du logiciel, un salon est caractérisé par une adresse IP multicast (Ex : `224.1.1.1`). Le port multicast est fixé à 5000 pour tous les noeuds. L'adresse multicast du salon est indiquée en ligne de commande au lancement du logiciel par l'option `-room`. Le pseudonyme de l'utilisateur est indiqué par l'option `-name`.
Vous trouverez ci-dessous des exemples portant sur les éléments : Socket en multicast, Thread.

1. Exemple d'utilisation de socket en réception

```
recvsock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
recvsock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
recvsock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEPORT, 1)
recvsock.bind(("224.1.1.1", 5000))
mreq = struct.pack("=4sl", socket.inet_aton("224.1.1.1"), socket.INADDR_ANY)
recvsock.setsockopt(socket.IPPROTO_IP, socket.IP_ADD_MEMBERSHIP, mreq)

recvsock.settimeout(1)
while True:
        try:
                print recvsock.recvfrom(1024)[0]
        except socket.timeout:
                continue
```

2. Exemple d'utilisation de socket en émission

```
sendsock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
sendsock.sendto("Mon message", ("224.1.1.1", 5000))
```

3. Exemple de Thread

```
class MyThread(Thread):
        def __init__ (self, arg1):
                Thread.__init__(self)
                ...

        def run(self):
                while True:
                        ...

mythrd = MyThread("Ceci est l'argument 'arg1'")
mythrd.start()
```

**1)** Compléter l'ébauche `chat-multicast.py` disponible depuis la page web du module. Lorsqu'un participant arrive et quitte le salon, il envoie un message d'annonce au groupe.

**2)** Déterminer l'adresse MAC multicast pour ce groupe ? Vérifier que l'adresse IP et MAC multicast sont bien ajoutées à l'interface réseau. Pour cela utiliser la commande `netstat -g` une fois l'application en cours d'exécution.

**3)** Effectuer une capture des échanges quand l'application démarre et pendant son fonctionnement. Quelle est l'adresse MAC utilisée en adresse de destination ?

**4)** Refaire cette application avec des échanges effectués en IPv6. La documentation associée aux sockets en IPv6 est décrite dans le rfc3493. L'adresse Multicast IPv6 utilisée sera `ff1x::0801:101` avec x la portée de l'adresse (rfc 3307, 4291).

### Exercice 4.2 — Multicast applicatif

Il s'agit d'arriver au même fonctionnement que dans la section précédente mais en s'appuyant cette fois-ci sur un mécanisme de multicast applicatif. Le mécanisme est une variante de Host Multicast Tree Protocol (HMTP) [1]. Cette partie donnera lieu à la réalisation de deux programmes : `rdvpoint.py` et `mcastnode.py`.

Le programme `rdvpoint.py` met en oeuvre la logique du point de Rendez-vous (RP), à savoir :
- L'enregistrement de la racine de l'arbre de distribution multicast.
- L'envoi de l'adresse IP et le port de la racine suite à une requête d'un noeud.
- Le vieillissement de l'enregistrement de la racine.

Le programme `mcastnode.py` traduit les fonctions attendues sur un noeud :
- Requête portant sur l'adresse IP et le port de la racine vers le RP.
- Parcours de l'arbre depuis la racine pour tenter de s'y rajouter.
- Tentative de rattachement à un noeud.
- Maintenance du rattachement au noeud parent par rafraichissement régulier.
- Réponse à une demande de rattachement d'un noeud.
- Maintenance du rattachement des noeuds enfants par temporisateur.
- Interfaçage entre les programmes utilisateurs et le réseau multicast (effectué à travers une communication socket depuis le programme utilisateur).
- Envoi des messages à partir de la racine.
- Affichage des messages reçus à travers le réseau multicast sur la sortie standard (la console).

L'adresse IP et le port du RP sont connus de tous les noeuds. Toutes les communications se font à travers des sockets UDP. Le RP écoute les requêtes et les demandes d'enregistrement de la racine sur le port MCAST_PORT=10000. Les noeuds s'échangent la signalisation pour la gestion de l'arbre et les données par le port MCAST_PORT. Ils reçoivent les données à envoyer issu du programme utilisateur par le port DATA_PORT=10001. Les échanges se présentent sous forme d'un message constitué d'un type (sur 4 caractères) et d'un corps (peut être vide ou non), séparés par un retour à la ligne.

La procédure de découverte et d'enregistrement de la racine de l'arbre de distribution multicast est représentée par la figure 1 et se décrit de la manière suivante :
- Une requête portant sur la racine consiste en un type ROOT envoyé au RP. La réponse est un message TOOR dont le corps est constitué de l'adresse IP et le port de la racine séparés par un espace. Le corps est vide lorsqu'aucun noeud n'est déclaré en tant que racine.
- Une demande d'enregistrement en tant que racine de type RREG est émis par un noeud lorsqu'aucune racine n'est déclarée et qu'il n'est pas encore rattaché à un parent. La réponse peut être YREG ou NREG selon que l'enregistrement réussi ou non. En cas de réponse négative, le noeud demandeur refait une demande de type ROOT pour découvrir le nouveau noeud racine. Si l'enregistrement réussit, le RP

mémorise l'adresse IP et le port de la racine et y associe un temporisateur. La racine doit rafraîchir régulièrement son enregistrement auprès du RP par l'émission d'un message RREG toutes les 2 minutes. En l'absence de message RREG de la part du noeud racine courant, le RP supprime l'enregistrement à l'expiration d'un temporisateur avec un délai de garde de 3 minutes.

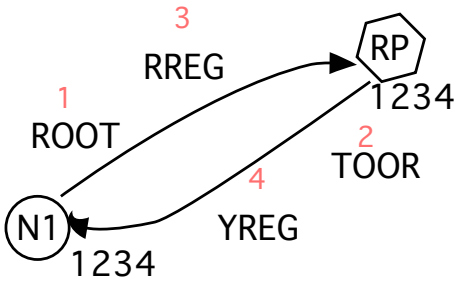


FIGURE 1 – Enregistrement de la racine.

La procédure de rattachement d'un noeud à l'arbre de distribution est indiquée ci-dessous et représentée par la figure 2 :

– Une demande de rattachement comme noeud enfant est du type RJOI. La réponse peut être YJOI ou NJOI selon que le rattachement réussi ou non. Le rattachement est en échec quand le noeud parent a atteint le nombre limite de noeuds enfants. Un message NJOI est suivi de la liste sous forme de lignes composées de l'adresse IP et du numéro de port des enfants de son émetteur. Le noeud recevant NJOI choisit aléatoirement un noeud parmi les noeuds enfant mentionné dans le message NJOI. Une nouvelle demande est soumise à ce noeud. Cette procédure est répétée jusqu'à ce que le nouveau noeud reçoit une réponse positive YJOI et est rattaché à l'arbre. Le noeud parent ajoute une entrée avec un temporisateur dans la table de ses enfants pour le nouveau noeud. Un noeud enfant rafraîchit régulièrement l'entrée de son parent (selon une période de 2 minutes). L'entrée est enlevée de la table en cas de non réception de message RJOI après l'expiration du temporisateur avec un délai de garde de 3 minutes. Un noeud enfant qui ne reçoit pas 3 fois de suite, une réponse de son parent le considère comme étant inaccessible et relance la procédure de rattachement à partir de la racine.
– Tout noeud a une limitation sur le nombre de noeuds enfants rattachés égal à CHILDREN_MAX=3. La demande de rattachement suit le même schéma que HTMP à la différence que le choix du parent (celui auquel la demande de rattachement est destinée) parmi les candidats se fait de manière aléatoire.

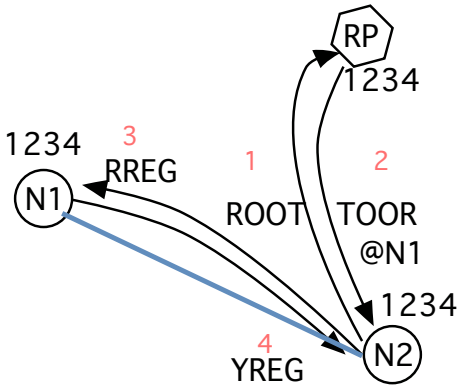


FIGURE 2 – Rattachement d'un noeud à l'arbre.

Les messages à distribuer sur l'arbre multicast sont envoyés à la racine (après demande de l'adresse et le port de la racine au RP). La racine affiche le message sur la sortie standard et le recopie vers ses noeuds enfants. Les noeuds réalisent la même opération jusqu'à ce que l'ensemble du groupe reçoit la donnée. L'adresse et le port de la racine obtenues d'une requête précédente, peuvent être mémorisés par les noeuds. La durée de validité de cette information doit être toutefois limitée à 1 minute

Les programmes `nc` et `iperf` peuvent être utilisés pour la transmission du message sur le port DATA_PORT du noeud localement. Ainsi lorsque les données proviennent de l'adresse localhost, elles sont propagées à la racine pour distribution.

A l'issu de ce sujet de TP, vous devez rendre le rapport des travaux pratiques accompagné des fichiers sources Python.

## Bibliographie

[1] **Zhang, B., Jamin, S. and Zhang, L.**. (2002). INFOCOM.
`https://www.cs.arizona.edu/~bzhang/paper/02-infocom-hmcast.pdf`
*Host Multicast : A Framework for Delivering Multicast to End Users.*