TP N°7 : Topologie en Mesh

1. Objectif

Dans ce TP, vous allez mettre en place un réseau maillé (*mesh*) en utilisant l'API BSD sockets mode UDP.

2. Principe de la topologie

Dans un réseau maillé (*mesh*), les nœuds sont tous connectés entre eux. Sans faire appel à une entité intermédiaire comme un switch ou routeur, dans le cas d'une topologie en étoile.

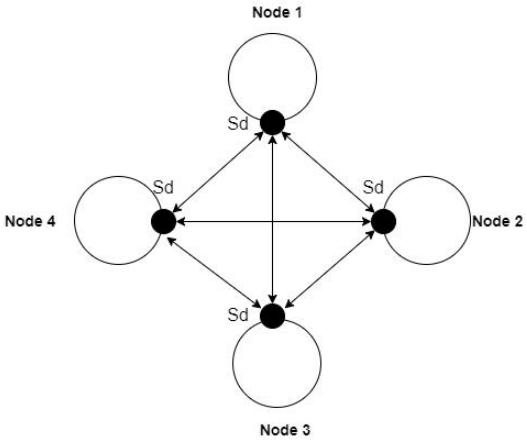
Même si le flux est bidirectionnel avec UDP, il sera utilisé, dans ce TP, uniquement dans un seul sens

3. Conception

3.1. Définition:

Dans cette TP, on utilise les sockets en mode UDP. Chaque nœud possède un seul socket pour l'entrée et sortie. Le but est de simplifier l'implémentation.

C'est faisable de faire appel aux socket en mode TCP, mais cela peut être compliquer pour certains lors de l'implémentation.

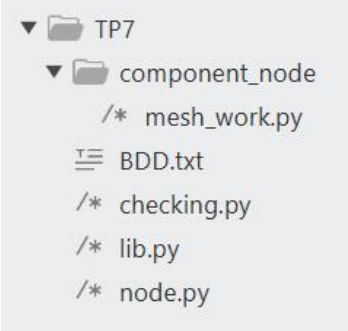


- La partie qui gère le socket est nommée **Sd**

3.2. Structure du TP

Dans ce TP, vous allez travailler aussi avec des **modules** (scripts) que vous allez implémenter vous-même.

On crée un dossier **component_node** dans lequel on place le script **mesh_work.py**



4. Implémentation

4.1. Module lib:

Étape 1: Ouvrir un éditeur de texte et enregistrer le fichier sous le nom l**ib.py**
Étape 2: Importer les modules nécessaire**s: socket, threading, sys.**

4.2. Module node

Ce module est le module principale qu'on va exécuté. Soit:

```
>python node.py 60000
```

- **60000:** représente le port pour le flux entrant et sortant

Étape 1: Ouvrir un éditeur de texte et enregistrer le fichier sous le nom **node.py**

Étape 2: Importer les modules :

```
1  from lib import *
2  from checking import *
3  from component_node.mesh_work import *
```

On remarque un module nommé «checking» qui vérifie le de «port» saisi comme argument; vérifier dans un fichier texte, nommé «BDD.txt», si le numéro de port n'est pas déjà enregistré et enregistrer le numéro de port dans «BDD.txt».

Étape 3: Récupérer le paramètre d'entrée contenant la valeur du numéro de port

```
PORT=int(sys.argv[1])
```

Étape 4: Récupérer les numéros de port enregistrés dans «BDD.txt» et les enregistrer dans une structure de données de type liste et nommée **my_list**.

```
creat_my_list('BDD.txt')
```

La structure **my_list** est initialisée dans le module **lib** . Dans la mesure ou les autres modules feront appel à **lib**, ils pourront eux aussi accéder à la structure **my_list** sans problème et la modifier.

On considère, ainsi, **my_list** comme variable globale.

La fonction **creat_my_list** est définie dans le module **checking.py**

```
def creat_my_list(name):
    with open(name, "r", encoding='utf-8') as f_name:
        all_lines = f_name.readlines()
        # Itérer sur les lignes
        for line in all_lines:
            my_list.append(line.strip())
```

Étape 5: Créer l'objet **sd** instancié de la classe **ConnectService**

Cette classe est définie dans le module **mesh_work**

```
sd=ConnectService(PORT)
```

Étape 6: Faire appel à la fonction **introduce_self** qui permet au socket définie dans l'étape 5 d'envoyer un message pour se faire connaître aux prêts des autres sockets dont le numéro de port est déjà enregistré dans «BDD.txt»

La fonction **introduce_self** est définie dans le module **mesh_work**

```
introduce_self(sd.s)
```

Étape 7: Lancer un thread qui s'occupe de la recéption des messages

```
th1=threading.Thread(target=handle_reception, args=(sd.s,))
th1.start()
```

La fonction **handle_reception** est définie dans le module **mesh_work**

Étape 8: Faire appel à la fonction **work_to_do** qui donne la main à l'utilisateur pour envoyer ou afficher des messages.

```
work_to_do(sd.s)
```

La fonction **work_to_do** est définie dans le module **mesh_work**

4.3. Module mesh_work:

Étape 1: Créer un dossier «component_node».
Étape 2: Entrer dans ce dossier
Étape 3: Ouvrir un éditeur de texte et enregistrer le fichier sous le nom **mesh_work.py**

Étape 4: Dans ce module, on défini la structure d'une classe **ConnectService** :

```
1   from lib import *
2   class ConnectService:
3       def __init__(self, port):
4           #créer socket en mode UDP nommée "s" (self.s)
5           ...
6           try:
7               #attacher socket "s" à une adresse "localhost" et numéro de port récupérer dans comme paramètre
8               ...
9           except:
10              print("Le Sd n'arrive pas à s'attacher à l'adresse & numéro de port")
11              sys.exit()
```

Étape 5: Définir la fonction **introduce_self :**

```
def introduce_self(s):
    #récupérer l'adresse de socket créée
    my_addr=s.getsockname()

    #le socket enverra un message de type string de la forme suivante:
    #"NEW:numéro de port
    msg=msg="New:"+str(my_addr[1])

    #parcourir la liste contenant tous les numéros de ports enregistrés et leurs envoyer le message précédent
    for i in range (len(my_list)):
        if my_list[i]!='':
            s.sendto(msg.encode(), ('localhost',int(my_list[i])))
```

Étape 6: Définir la fonction **handle_reception:**

```
def handle_reception(s):
    while True:
        #recevoir un message et le stocker dans "msg"
        ...

        #Verifier la nature du message reçu
        #Si le message contient la chaîne de caractère "New"
        # alors il s'agit d'un message provenant d'un noeud nouveau, donc il faut l'enregister dans "my_list"
        # l'enregistrement se fera dans une fonction à part nommée "discover"
        if ...:
            discover(msg.decode())
        else:
        #Sinon il s'agit d'un simple message qu'il faut l'afficher
            ...
```

Étape 7: Définir la fonction **discover :**

```
def discover(msgD):
    msg1=msgD.split(":")
    my_list.append(int(msg1[1])) # récupérer le numéro de port du nouveau noeud
```

Étape 8: Définir la fonction **work_to_do :**

```
def work_to_do(s):
    while True:
        print("----------------------------------------")
        print("Vous avez la main de faire quelque chose:")
        print("   1: Envoyer un message:")
        print("   2: Afficher la liste de vos voisins")
        choice=input("Choix ? : ")
        if choice== "1":
            x=input("Entrer le message : ")
            y=input("Entrer le destinataire : ")
            s.sendto(x.encode(),('localhost',int(y)))
        if choice== "2":
            print(my_list)

#L'instruction match n'existe pas dans la version 3.7
```

5. Exécution

Ouvrir 03 terminaux, lancer le programme dans chaque terminal.

```
>python node.py 1001
>python node.py 1002
>python node.py 1003
```