"Mini-chat" simple sur réseau

Résumé : fiches d'activités sur ordinateur pour comprendre l'adressage IP, et la mise en œuvre basic du protocole TCP/IP.

Thématique : Localisation, cartographie et mobilité

Point du programme traité :

Contenus : Protocole TCP/IP

Capacités attendues : Identifier l'adresse IP de plusieurs machines. Envoyer et recevoir simplement des messages textuels entre deux ordinateurs connectés au réseau local via le protocole TCP/IP.

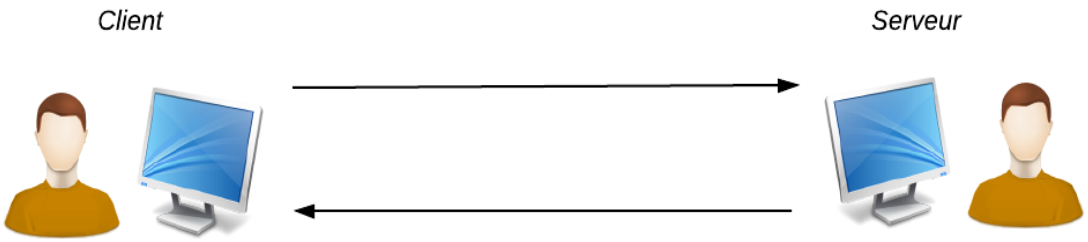
Lieu de l'activité : En salle informatique

Matériels / logiciels utilisés : Ordinateur connecté en réseau / utilisation de Python

Durée de l'activité : 1h environ

Conditions : Comme le niveau de sécurité des salles informatiques ne semble pas identique dans chaque lycée de l'académie, il est impératif de vérifier auprès du responsable informatique, avant la séance que l'on accède bien à l'invite de commande en ligne et que le port 3222 est autorisé pour le transfert de données.

Principe de la communication

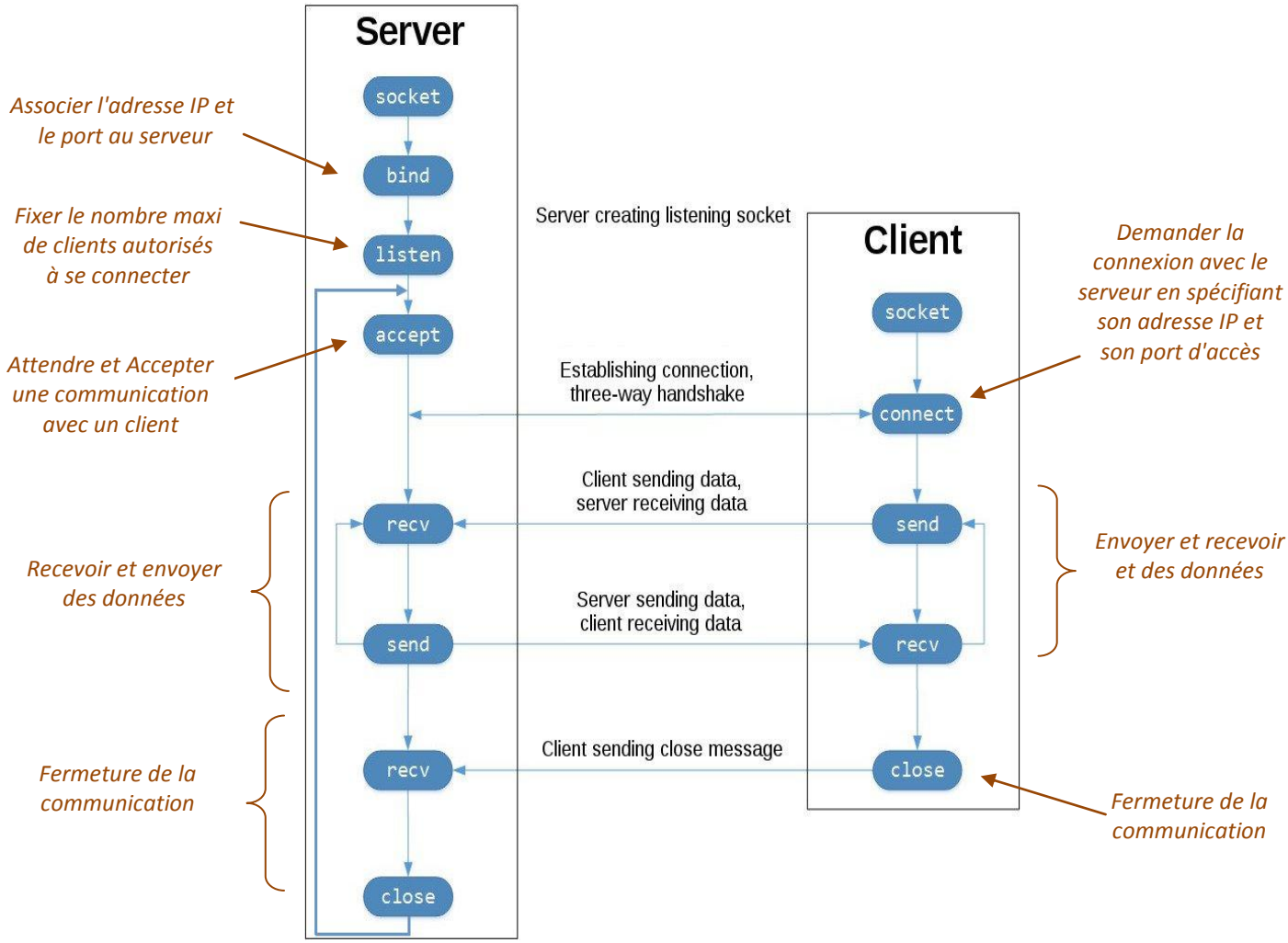


Pour la réalisation du "minichat", on utilise le principe de la communication client / serveur qui la base de toutes communications des objets connectés sur le réseau TCP/IP.

- Le serveur attend une communication pour cela il scrute en permanence les entrées, affiche les messages reçus et retourne une réponse si nécessaire.

- Le client initie la communication en spécifiant l'adresse IP et le port du serveur. si la communication est acceptée, le client envoie et reçoit les messages au serveur. Lorsque l'échange est terminé, le client ferme la communication.

Algorithme graphique de fonctionnement et échange de données

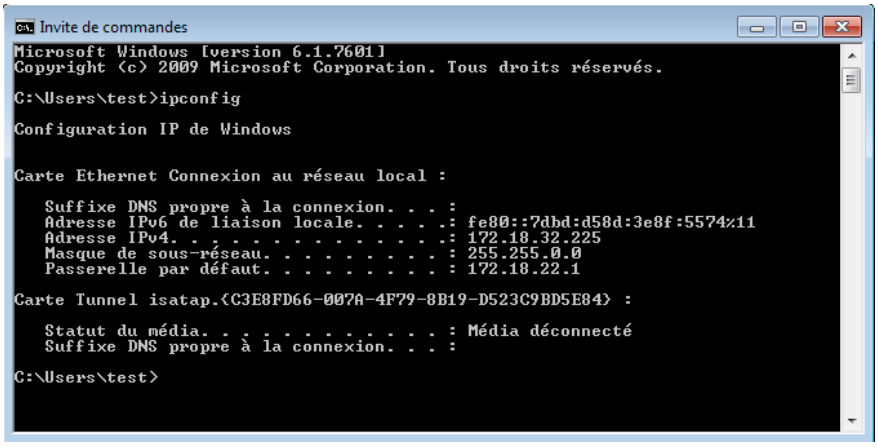


Travail à faire

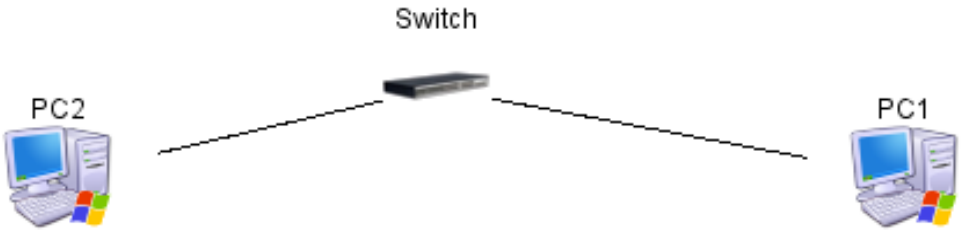
Identification des ordinateurs

1) Choisir son binôme définir le rôle client et serveur

2) A partir de la commande ipconfig trouver l'adresse IP de l'ordinateur client et celle de l'ordinateur serveur



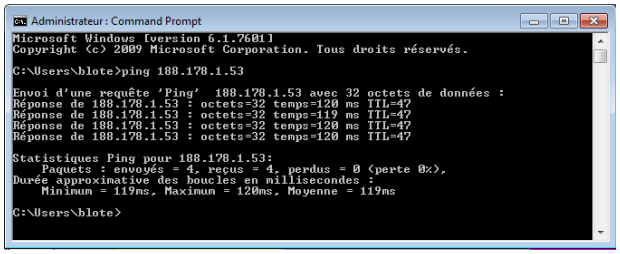
3) Compléter le schéma

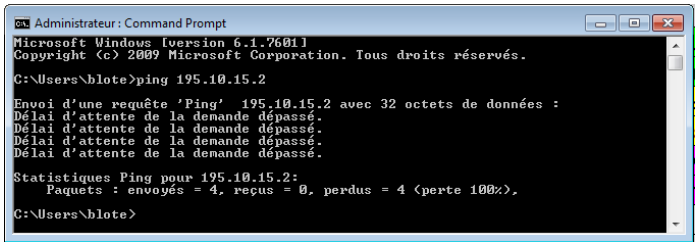


Nom de l'utilisateur :
rôle :
adresse IP :
masque de sous réseau :
Passerelle :

Nom de l'utilisateur :
rôle :
adresse IP :
masque de sous réseau :
Passerelle :

4) A l'aide de la commande ping adresseDestinataire, vérifier la communication





5) Laquelle de ces deux fenêtres montre que la liaison réseau est établie

Emission d'un message sur l'ordinateur 1,

6) Saisir le programme "client.py" qui permet d'envoyer un texte à l'ordinateur distant disposant du programme serveur

```
import socket

# programme le plus simple possible
s = socket.socket()                     # création de l'instance
s.setblocking(False)                    # en mode non bloquant c'est mieux
s.connect( ("172.18.132.225", 3222) )   # spécifie l'adresse IP du serveur via le port 3222
s.sendall("bonjour Philippe")           # envoyer le message bonjour au destinataire

s.close()                               # fermer la communication quand on a terminé
print("fin de la communication")
```

Réception d'un message sur l'ordinateur 2

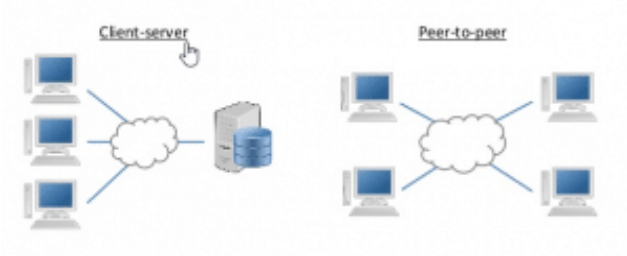
7) Saisir le programme "serveur.py" qui scrute le réseau et établi la connexion avec le client demandeur.

```
import socket

# programme le plus simple possible
s = socket.socket()
s.bind( ("172.18.132.225", 3222) )                # identifier le serveur 172.18.132.225 via le port 3222
s.setblocking(False)                              # en mode non bloquant c'est mieux pour cette demo
s.listen(1)                                       # écouter 1 connexion entrante à la fois
while True :
   try :
      clientConnected, client_address = s.accept()  # récupere l'adresse du client connecté
      print( str(client_address)+" connecté." )     # affiche l'adresse du client distant
      print( clientConnected.recv(32) )             # affiche les caractères reçus

      clientConnected.close()                       # on se déconnecte quand on a terminé
   except :  pass
print("fin de la communication")
```

8) Lancer l'exécution du serveur puis le client. Vérifier que le message est bien transmis.

9) Tester le fonctionnement avec un autre interlocuteur

10) Sachant que la communication est limitée à deux interlocuteurs à la fois. Conclure sur le mode de communication.



Ajouter une réponse dans la communication

11) dans le programme client.py, après avoir envoyer le message "bonjour Philippe", ajouter la ligne de code suivante qui permet d'afficher une éventuelle réponse du serveur.

```
print( s.recv(32) )                    # affiche la reponse reçu
```

12) dans le programme serveur.py, après avoir affiché les caractères reçus, ajouter la ligne de code suivante qui permet d'envoyer le message "bonjour Eric" au client.

```
clientConnected.sendall(b"bonjour Eric")      # répondre au message
```

13) Lancer l'exécution du serveur puis le client. Vérifier que les messages sont transmis dans les deux sens.

Amélioration du chat (facultatif)

14) Proposer un proposer un algorithme simple pour gérer le client et le serveur dans le même programme en ayant la possibilité de saisir un texte.

15) Intégrer le client et le serveur dans le même programme puis ajouter la fonction input pour saisir le texte à envoyer.