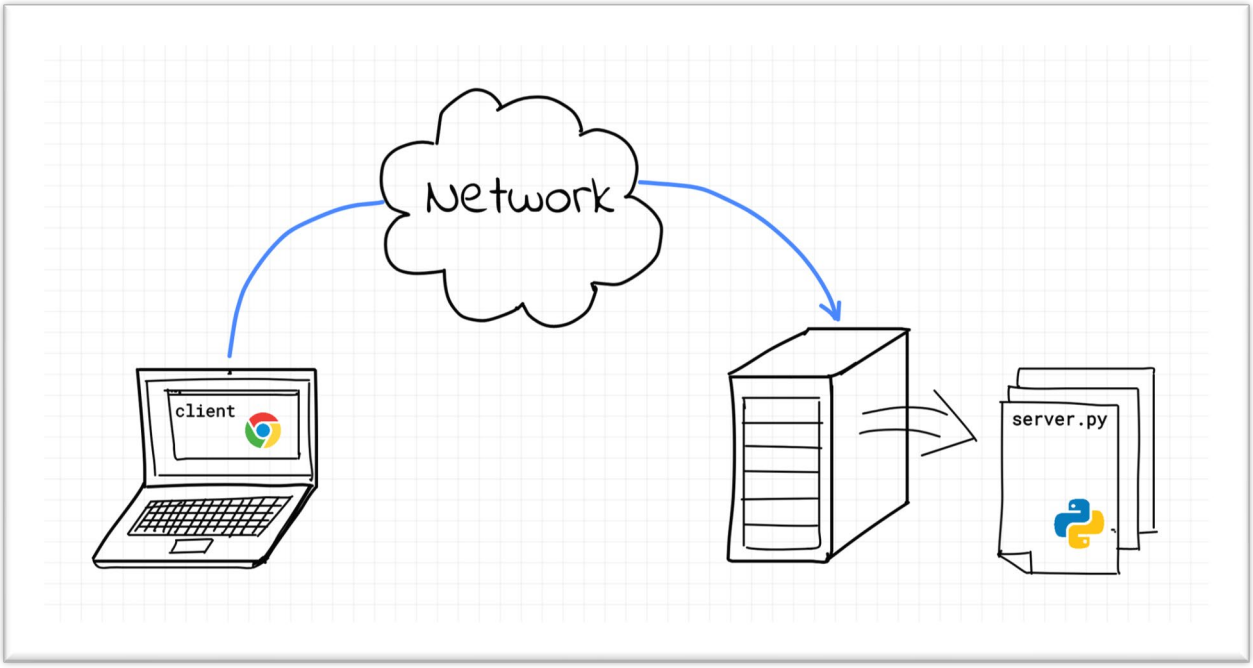
TP Socket TCP/IP – Architecture N-tiers- Bachelor 3 CDA Groupe B- 2024



Objectifs du TP : 2
Matériel Nécessaire (Pour rendu python) : 2
Théorie 2
Socket TCP/IP 2
Architecture Client-Serveur 3
Les étapes d'une communication Client-Serveur 4
Établissement de la connexion 4
Ressources utiles pour le travail pratique : 5
Suivi et rendu du travail pratique : 5
Travail pratique (noté) : 5
Partie Serveur 5
Partie client 6

Objectifs du TP :

- Comprendre et mettre en œuvre l'architecture client-serveur.
- Pratiquer la programmation de sockets en Python ou PHP pour la communication réseau.
- Appliquer les concepts de base de l'échange de données entre un client et un serveur.

Matériel Nécessaire (Pour rendu python) :

- Votre PC avec Python installé (Python 3.x recommandé).
- Accès à un éditeur de texte et à un terminal ou un environnement de développement intégré (IDE) pour écrire et exécuter des scripts Python.

Théorie

Socket TCP/IP

Chaque machine (hôte) au sein du réseau informatique de l'EPSI doit être identifiée par une adresse unique. Par exemple, 192.168.76.5 pourrait représenter une adresse IP au sein du réseau de l'école. Pour simplifier la gestion et l'accès aux machines, il est possible d'attribuer des noms symboliques à ces adresses IP. Ainsi, sur le réseau de l'EPSI de Lille, des noms comme Bacasable1 ou MyDil23 pourraient être associés à des adresses IP des PC utilisés dans les salles. Ces adresses, étant privées, ne sont accessibles que depuis le réseau interne de l'école.

En outre, chaque machine est équipée de ports logiques, numérotés de 1 à 65535, qui distinguent les différents services proposés (tels que la messagerie, le web, etc.). Une socket, ou "prise" en français, constitue un point de communication qui permet à un processus d'envoyer et de recevoir des données. La socket est définie par un numéro unique et fonctionne comme un descripteur de fichier, facilitant la manipulation des flux de données.

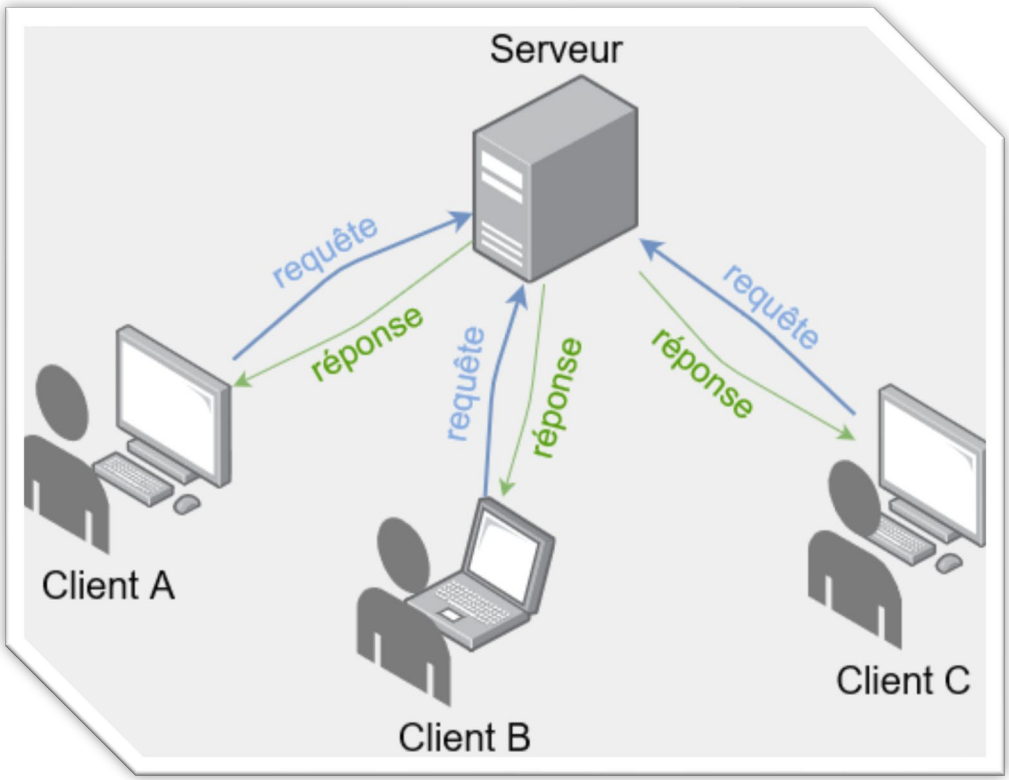
Dans le contexte du protocole TCP, pour qu'un échange d'informations puisse se dérouler, il est nécessaire que la socket soit connectée à une autre socket. Cette infrastructure permet aux étudiants et au personnel de l'EPSI de bénéficier d'un système de communication réseau adapté.

Architecture Client-Serveur

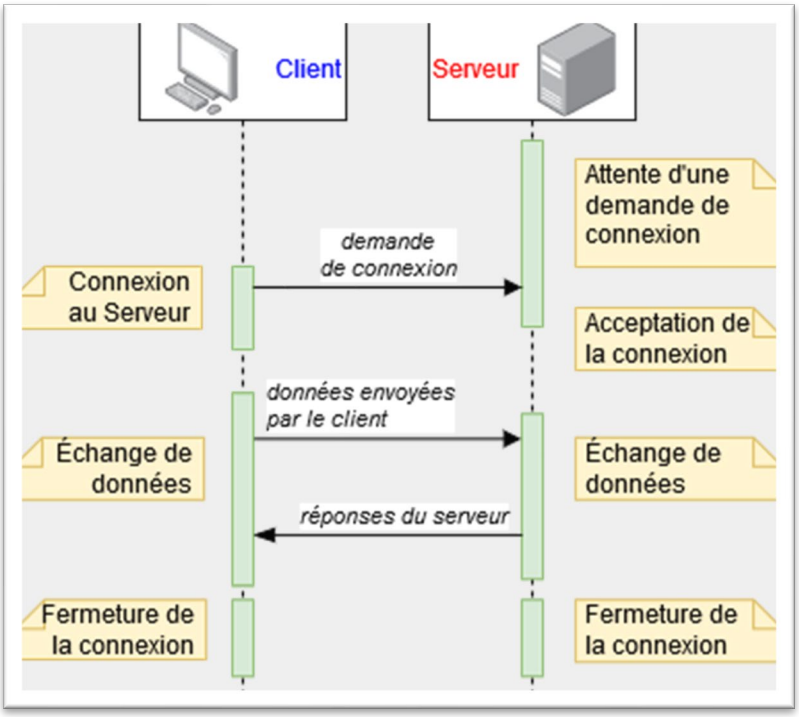
Dans l'**architecture Client-Serveur**, on trouve en général un serveur et plusieurs clients.
Le **serveur** est une machine qui traite les *requêtes* du **client** et lui envoie éventuellement une *réponse*.

Il y a donc deux types d'application installés sur les machines :

- L'application « **serveur** » : *écoute* en attendant des connexions des clients ;
- Les applications **« client** » : se *connectent* au serveur



Les étapes d'une communication Client-Serveur



Le *serveur* :

1. Attend une connexion de la part du client ;
2. Accepte la connexion quand le client se connecte ;
3. Échange des informations avec le client ;
4. Ferme la connexion.

Le *client* :

1. Se connecte au serveur ;
2. Échange des informations avec le serveur ;
3. Ferme la connexion.

Établissement de la connexion

Pour que le *client* se connecte au *serveur*, il lui faut deux informations :

- L'**adresse IP** du serveur, ou son **nom d'hôte** (*host name*), qui identifie une machine sur Internet ou sur un réseau local.

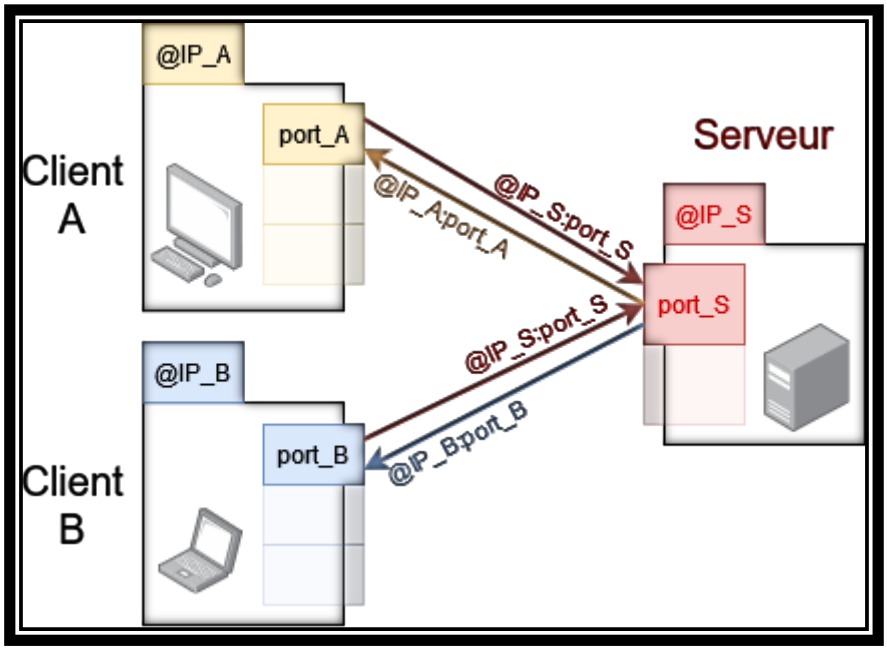
Les noms d'hôtes permettent de représenter des adresses IP de façon plus claire
par exemple 'google.fr' est plus facile à retenir que l'adresse IP correspondante 74.125.224.84

- Le **numéro de port** associé à l'application qui doit gérer les requêtes sur le serveur.

Pour que le *serveur* réponde au *client*, il lui faut également deux informations :

- Le **nom d'hôte** (*host name*), du client ;

- Le **numéro de port** associé à l'application qui doit recevoir les réponses sur le client.



Ressources utiles pour le travail pratique :

https://docs.python.org/fr/3/howto/sockets.html Documentation python sur les sockets TCP/IP.

https://www.baeldung.com/a-guide-to-java-sockets Idem mais pour Java.

Effectuez également vos propres recherches sur les différents protocoles et commandes utilisés.

Suivi et rendu du travail pratique :

Note Johann : Le travail suivant sera noté individuellement.

Il faudra m'envoyer les 2 scripts que vous aurez créé à l'adresse johann.devred@campus-cd.com | Votre nom et prénom, suivi d'un lien web vers une photo d'une personnalité célèbre (vivante, décédée ou fictive) **qui porte le même prénom que vous**, dans une ligne #commentée dans vos **2** scripts 😉.

Vous êtes également libres de me rendre ce travail au langage PHP ou java si vous le souhaitez, en gardant les mêmes consignes de rendu.

Plus les scripts seront commentés, meilleures seront vos notes. Bon courage !

Deadline de rendu : Mardi 05 mars 13h30

Travail pratique (noté) :

Partie Serveur

Composez un script serveur en Python intitulé « **Serveur_votreprenom.py** » pour accomplir les opérations ci-dessous :

- Instancie un socket d'écoute et le lie à un numéro de port déterminé par votre âge. Utilisez 50000 comme point de départ et ajoutez votre âge pour obtenir le port. Dans l'éventualité où le port sélectionné est déjà occupé sur votre système, augmentez-le de 100 jusqu'à ce qu'un port disponible soit trouvé.
- Active le mode d'écoute du socket.
- Affiche l'adresse du client dès sa connexion.
- Attend de recevoir un message du client. Renvoie immédiatement ce message au client, en y ajoutant "\nJe suis là !" à la fin.
- Clôture les connexions après l'échange.

Partie client

Créer un script en Python **« Client_votreprenom.py »** qui :

- Initialise un socket.
- Établit une connexion avec le serveur en utilisant l'adresse localhost suivi du numéro de votre port. Dans ce contexte, le client et le serveur fonctionnent sur le même dispositif (votre ordinateur).
- Transmet la requête "Serveur es-tu là, tu vas bien, je m'appelle (**votre prénom**) ?" au serveur.
- Affiche le retour envoyé par le serveur.
- Interrompt la connexion une fois l'échange terminé.

- Exécutez d'abord le script serveur et ensuite le script client dans deux terminaux différents.
- Observez la communication entre le client et le serveur, et notez la réponse reçue par le client.
- Assurez-vous de bien fermer les sockets dans les deux scripts.
- Modifier le programme de manière que le client termine la connexion lorsque l'utilisateur envoie le texte indiqué sur l'image suivante :

Détaillez les différentes étapes effectuées sur un document word/pdf à part à m'envoyer en fichier joint.

Bonus (facultatif) : Ajoutez une fonctionnalité permettant au client d'envoyer des commandes spécifiques au serveur (par exemple, **date** pour recevoir la date et l'heure actuelle du serveur). Ces fonctionnalités devront être #commentées.

Bonus ++ (facultatif) : Intégrer une interface graphique fonctionnelle coté client.

« J'ai fini 😊, je veux rendre mon TP » : voir ici