Internet et bases de données
M2SOL021 « Langue et Informatique »
TD05 : Sockets et multithreading
Ana ARMENTA. 28503310

Pre-Rapport

Sommaire

1 Pré-Rapport

1.1 Quelle est la différence entre un processus et un thread ?

Les processus et les threads sont des séquences d'exécution indépendantes. La principale différence est : Les threads (du même processus) s'exécutent dans un espace mémoire partagé, tandis

que les processus s'exécutent dans des espaces mémoire différents. Un programme en cours d'exécution est souvent appelé processus. Un thread est un sous-ensemble du processus. KOUIS 2018

	Processus	Thread
Définition	Un programme en cours d'exécution s'appelle un processus.	Un thread est une petite partie d'un processus.
La communication	La communication entre deux processus est coûteuse et limitée.	La communication entre deux threads est moins coûteuse que celle du processus.
Multitâche	Le multitâche basé sur les processus permet à un ordinateur d'exécuter deux ou plusieurs programmes simultanément.	Le multitâche basé sur les threads permet à un programme unique d'exécuter deux threads ou plus simultanément.
Espace d'adressage	Chaque processus a son espace d'adressage distinct.	Tous les threads d'un processus partagent le même espace d'adressage que celui d'un processus.
Tâche	Les processus sont des tâche lourde.	Les threads sont des tâches légères.
Exemple	Vous travaillez sur un éditeur de texte, il fait référence à l'exécution d'un processus.	Vous imprimez un fichier à partir d'un éditeur de texte tout en travaillant dessus, ce qui ressemble à l'exécution d'un thread dans le processus.

FIGURE 1 – Table de comparaison

La figure 1 a été prise du site web WayToLearnX [1]

— Un processus se compose de plusieurs threads. Un thread est la plus petite partie du processus pouvant s'exécuter simultanément avec d'autres parties (threads) du processus.

— Un processus est parfois appelé tâche lourde. Un thread est souvent appelé processus léger.

— Un processus a son propre espace d'adressage. Un thread utilise l'espace d'adressage du processus et le partage avec les autres threads de ce processus.

1.2 Quelle est la différence entre les protocoles TCP et UDP et à quelle couche du modèle OSI appartiennent-ils ?

Deux grands protocoles se partagent la majorité des opérations sur la couche Transport. Ces protocoles sont TCP et UDP.

1. https ://waytolearnx.com/2018/09/difference-entre-thread-et-processus-en-java.html

Vaucamps 2009

TCP "Transmission Control Protocol", offre des services d'établissement et de fin de dialogue ainsi que des messages de maintenance de la communication en mode fiable et connecté avec :

— des accusés de réception
— du séquençace, de l'ordonnancement
— du contrôle de flux (fenêtrage)
— de la reprise sur erreur
— du contrôle de congestion
— de la temporisation

UDP "User Datagram Protocol", s'occupe uniquement du transport non fiable.

Il est une simple passerelle entre IP et l'application. Il est conseillé pour les applications pour du trafic en temps réel à taille fixe et régulier (voix, vidéo). Il supporte des protocoles simples (TFTP, SNMP) ou souffrant des délais (DHCP, DNS, NTP). Les paragraphes ci-dessus ont été inspirés par le site Web cisco.goffinet.org. [2]

Différence entre TCP et UDP KOUIS 2017

— TCP est en mode orienté connexion et fiable, tandis que UDP est en mode non-connecté et peu fiable.
— TCP nécessite plus de traitement au niveau de l'interface réseau, ce qui n'est pas le cas en UDP.
— TCP utilise le contrôle de la congestion, le contrôle de flux et d'autres mécanismes pour assurer la transmission fiable.
— UDP est principalement utilisé dans les cas où le retard de paquet est plus sérieux que la perte de paquet.

1.3 Qu'est-ce que un socket ?

[3] Un socket correspond à un identifiant de processus dans un environnement réseau TCP/IP.

2. https ://cisco.goffinet.org/ccna/ipv4/couche-transport-tcp-et-udp/
3. Cette réponse a été inspirée par le livre Dordoigne 2011

L'identifiant de socket est composé de l'adresse IP, du protocole de couche 4 OSI utilisé (TCP ou UDP), ainsi que d'un numéro de port (TCP ou UDP). Ce numéro de port est compris entre 0 et 65535.

Socket = adresse IP : TCP ou UDP : numéro de port

Par exemple, un socket serveur HTTP serait de la forme : 192.168.2.200 :TCP :80.

Un socket est dynamiquement alloué au client, afin de permettre une véritable communication, dans les deux sens.

1.4 Quelle est la relation entre les sockets et le multi-threading ?

Une socket est un point d'accès aux couches réseau TCP/UDP. les sockets constituent un mécanisme qui permet la communication sur le réseau Internet ou entre processus locaux tournant sur une même machine ;

Le multi-threading est une forme de parallélisation ou de division du travail pour un traitement simultané. Au lieu de donner une charge de travail importante à un seul cœur, les programmes threaded divisent le travail en plusieurs tâches (threads) logicielles. Ces tâches sont traitées en parallèle par différents cœurs de processeurs pour gagner du temps.*Qu'est-ce que l'hyper-threading ?* p. d.

1.5 Quelle est la différence entre un socket serveur et un socket client ?

La différence est que le client initie la connexion et n'écoute que les réponses tandis que le socket serveur écoute toujours et n'envoie que des réponses. KOUIS 2019

2 TD

2.1 Mini-chat TCP

Exercice 1 : Construction d'un serveur rudimentaire Nous mettrons en place un serveur capable de communiquer avec un seul client. Nous verrons un peu plus loin ce qu'il faut lui ajouter afin qu'il puisse prendre en charge en parallèle les connexions de plusieurs clients. La première version du serveur de notre Mini-chat comprend les 6 etapes suivantes : [4]

1. Création du socket import socket
   import sys
   HOST = socket.gethostname()
   print(HOST)
   IP = socket.gethostbyname(HOST)
   print(IP)
   PORT = 5001
   counter = 0
   mySocket = socket.socket(socket.AF_INET,
   socket.SOCK_STREAM)

2. Liaison du socket à une adresse précise ("socket serveur")
   try :
   mySocket.bind((IP, PORT))
   except :
   print("La connexion a echoué")
   sys.exit()

3. Attente de la requête de connexion d'un client while 1 :
   print("Serveur prêt")
   mySocket.listen()
   print("Je vous ecoute")

4. Malheureusement, je n'ai pas trouvé de moyen d'indenter le code

connexion, adresse = mySocket.accept()

4. Etablissement de la connexion
   counter += 1
   print("Client connecté, adresse IP : , port : ".format(counter, adresse[0], adresse[1], PORT))
   msgServeur = "Vous êtes connecté au serveur Sorbonne. Vous pouvez envoyer des messages"
   connexion.sendall(msgServeur.encode("Utf8"))

5. Dialogue avec le client
   while 1 :
   msgClient = connexion.recv(1024).decode("Utf8")
   if not msgClient or msgClient.upper()=="FIN" :
   break
   print("C> ".format(msgClient))
   msgServeur = input("S> ")
   connexion.sendall(msgServeur.encode("Utf8"))

6. Fermeture de la connexion
   print("Se cierra la conexion")
   connexion.close()
   ch = input("<R>ecommencer <T>erminer ?")
   if ch.upper() == "T" :
   break

Exercice 2 : Construction d'un client réseau rudimentaire

1. Création du socket
   import socket
   import sys
   HOST = socket.gethostname()

```
print(HOST)
IP_Serveur = socket.gethostbyname(HOST)
print(IP_Serveur)
PORT_Serveur = 5001
counter = 0
mySocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

2. Envoi d'une requête de connexion au serveur
```
("socket client")
try :
mySocket.connect((IP_Serveur, PORT_Serveur))
except socket.error :
print("La connexion a echoué")
sys.exit()
print("Se establecio la conexion con el servidor")
```

3. Dialogue avec le serveur
```
while 1 :
msgServeur = mySocket.recv(1024).decode("Utf8")
print("S> ".format(msgServeur))
msgClient = input("Vous >")
if msgClient.upper() == "FIN" :
break
mySocket.sendall(msgClient.encode("Utf8"))
```

4. Fermeture de la connexion
```
print("Se cierra la conexion")
mySocket.close()
```

Exercice 3 : Construction d'un serveur multithread
Nous mettrons en place un serveur capable de communiquer plusieurs clients en parallèle. Ce serveur n'est pas utilisé lui-même

pour communiquer : ce sont les clients qui communiquent les uns avec les autres, par l'intermédiaire du serveur. Celui-ci joue donc le rôle d'un relais : il accepte les connexions des clients, puis attend l'arrivée de leurs messages. Lorsqu'un message arrive en provenance d'un client particulier, le serveur le ré-expédie à tous les autres, en lui ajoutant au passage une chaîne d'identification spécifique du client émetteur, afin que chacun puisse voir tous les messages, et savoir de qui ils proviennent. [5]

1. Création d'objet thread pour gérer la connexion avec un client.
   Dialogue avec le client.
   Fermeture de la connexion.
2. Création du socket
3. Liaison du socket à une adresse précise ("socket serveur")
4. Attente de la requête de connexion des clients
5. Etablissement des connexions
6. Démarrage des threads pour gérer les connexion

Exercice 4 : Construction d'un client et d'un client réseau rudimentaire

1. Création d'objet thread pour gérer la réception des messages.
   Dialogue avec le serveur (réception de messages).
   Fermeture de la connexion.
2. Création d'objet thread pour gérer l'émission des messages
   Dialogue avec le serveur (émission de messages)
   Fermeture de la connexion
3. Création du socket
4. Envoi d'une requête de connexion au serveur ("socket client")

5. Le professeur a expliqué les exercices 3 et 4 et nous les avons testés ensemble.

5. Démarrage des threads pour gérer la réception et l'émission des messages

Références

[] *Qu'est-ce que l'hyper-threading ?* fr. URL : https://www.intel.com/content/www/fr/fr/gaming/resources/hyper-threading.html.

[Dor11] J. DORDOIGNE. *Recursos Informáticos Redes informáticas - Nociones fundamentales - [3a edición]*. Recursos informáticos. ENI, 2011. ISBN : 978-2-7460-6748-6. URL : https://books.google.fr/books?id=25Lmhq58C38C.

[KOU17] Amine KOUIS. *Différence entre les protocoles TCP et UDP*. fr-FR. Déc. 2017. URL : https://waytolearnx.com/2017/12/difference-entre-les-protocoles-tcp-et-udp.html (visité le 14/04/2021).

[KOU18] Amine KOUIS. *Différence entre Thread et Processus en Java*. fr-FR. Sept. 2018. URL : https://waytolearnx.com/2018/09/difference-entre-thread-et-processus-en-java.html (visité le 11/04/2021).

[KOU19] Amine KOUIS. *Différence entre socket client et socket serveur*. fr-FR. Mar. 2019. URL : https://waytolearnx.com/2019/03/difference-entre-socket-client-et-socket-serveur.html (visité le 14/04/2021).

[Vau09] A. VAUCAMPS. *Les réseaux avec Cisco : Connaissances approfondies sur les réseaux*. Ressources informatiques. Editions ENI, 2009. ISBN : 978-2-7460-4944-4. URL : https://books.google.fr/books?id=WbRgZCd18Z8C.