Guía de l’usuari

Ús del Python Script (ATENCIÓ: NOMÉS PER LINUX!!)

Instal·lació en Linux (Contacta amb els autors si vols suport per Windows):

Per tal d’executar el python script es requereix dels següents fitxers:

- UDPServer_Cornella-Ixent_Lupu-Roberto_entrega_final.py (python script)
- UDPClient_Cornella-Ixent_Lupu-Roberto_entrega_final.py (python script)
- fitxers (carpeta amb els fitxers que s’enviaran)
- venv (virtual environment)

Degut a que el nostre python script del Client requereix de la llibreria *inquirer* per tal de funcionar, proporcionem un *virtual environment.* D’aquesta manera no cal insta·lar la llibreria *inquirer* al vostre ordinador. Aquest pas es pot saltar si vostè vol instal·lar la llibreria *inquirer* al seu dispositiu. Però degut a que en els ordinadors de la EPSEVG no es pot fer, donem aquesta opció. Els passos a seguir són el següents:

1. En primer lloc cal activar el venv amb l següent comanda:

    $ venv/bin/activate

D’aquesta manera, tota comanda s’hauria de veure amb el següent format (noteu que surt (venv) per pantalla abans de cada comanda). Tota instrucció executada amb aquest format insta·larà i usara llibreries a dins de la carpeta venv, que es podrà borrar sempre que es vulgui sense deixar cap rastre. Sempre es podrà anul·lar l’efecte del venv ambv la següent comanda:

    $ deactivate

2. Instal·lar la llibreria inquirer al venv amb la següent comanda

    $ pip install inquirer

Nota: En cas de voler executar aquesta comanda sense el venv es requereix instal·lar pip si es que no es té instal·lat. Es pot fer amb la següent comanda:

    $ sudo apt install python3-pip

Arrencada:

Per tal d’arrancar el server es suficient amb executar la següent comanda:

```
$ python3 UDPServer_Cornella-Ixent_Lupu-Roberto_entrega_final.py
```

El Client, per altrea banda s’ha d’executar amb la següent comanda (important tenir el venv actiu amb la llibreria inquirer o bé tenir l’inquirer instal·lat al PC):

```
$ python3 UDPClient_Cornella-Ixent_Lupu-Roberto_entrega_final.py
```

Guia d’us del Client:

A l’hora d’executar el client es pot fer tranquil·lament, ja que tant si el server està ocupat com si encara no està obert és un problema del qual l’usuari no s’ha de fer càrrec, ja que s’encarrega el codi de vetllar pel correcte funcionament amb l’ajuda de les capçaleres que indiquen quin tipus de paquet és cadascun i quin és el procediment a realitzar davant de cadascun. De fet el funcionament permet una comunicació entre un servidor i client, mentre que un nou client intenta relalitzar la petició. De manera que mentre es fa aquesta transmissió (independenment de si es tracta d’un GET o un PUT) el nou client va enviant peticions limitades pel timeout fins que el servidor acaba la transmissioó i es pot ocupar del nou client.

El primer que es demanarà a l’executar el client serà el nom del fitxer. S’ha de posar un nom que estigui en la carpeta fitxers adjunta al client en cas que es tracti de un PUT, o en la carpeta de fitxers adjunta al server en cas de voler realitzar un GET. De totes formes el programa envia automàticament el paquet d’error en cas de tractar-se d’un arxiu inexistent que provocaria l’abortament en el cas del Client però mai en el cas del Servidor ja que aquest s’haura d’encarregar de futures peticions.

Després, es demana si es vol fer un PUT (WRQ) o un GET (RRQ) al servidor. Ho decideix el client.

En tercer lloc, es tracten les opcions, que estan formades pel tamany de MSS i el timeout. Es demanaran opcions fins que el client així ho decideixi. El Servidor respon amb el pertinent OACK, i comença la transmissió. Els paquets s’envien de 512 a 512 bytes per defecte, però es pot negociar.

En qualsevol moment es poden perdre paquets, per tant, hi ha possiblitat d’un timeout. En aquest cas la part que envia el fitxer tornarà a enviar el paquet perdut, mentre l’altra part que rep el fitxer espera el paquet fins que li arriba finalment.

Al final del programa es contabilitzen els paquets enviats, i també els paquets perduts i que s’han hagut de reenviar.

Cal destacar que el servidor pot rebre moltes peticions a l’hora i que es pot obrir fins i tot després de que el Client envii la petició sense deixar de funcionar.

Sobre la pràctica

A l'hora de fer la pràctica, hem seguit l'ordre progressiu de classe, és a dir, entrega1, entrega2, ... Fins a arribar a l'entrega final. Això sí, hem tingut alguns problemes, sobretot a l'hora de transmetre paquets, ja que de vegades alguns paquets s'enviaven dos cops, degut a un error de programació al timeout. A més, també hem tingut problemes amb el numero de sequencia sobretot, que hem resolt utilitzant «xivatos» , i al final, ens ha quedat el codi resultant.