Centre universitari adscrit a la

TecnoCampus

upf. Universitat Pompeu Fabra *Barcelona*

Xarxes i Serveis
Pràctica 2

Dr. Pere Tuset-Peiró
{ptuset}@tecnocampus.cat

Grau en Enginyeria Informàtica de Gestió i Sistemes d'Informació
Escola Politècnica Superior TecnoCampus
Universitat Pompeu Fabra

Curs 2023-2024

Atenció

Abans de començar la pràctica es recomana llegir amb deteniment aquest enunciat per entendre l'objectiu i els requisits de l'activitat, i també el document RFC 1350 [*The TFTP Protocol (Revision 2)*] per entendre el funcionament del protocol TFTP.

1. Introducció

En aquesta segona pràctica de l'assignatura ens centrarem en la implementació d'un RFC (*Request for Comments*), que són els documents que estandarditzen els diferents protocols d'Internet desenvolupats per la IETF (*Internet Engineering Task Force*). En particular, l'objectiu d'aquesta pràctica és implementar un client del protocol TFTP (*Trivial File Transfer Protocol*), definit al RFC 1350, utilitzant el llenguatge de programació Python. Així doncs, el vostre client TFTP haurà d'implementar tant l'opció de lectura (llegir un fitxer ubicat al servidor remot) com l'opció d'escriptura (escriure un fitxer local al servidor remot).

Per tal de validar el correcte funcionament de la vostra implementació de l'estàndard, heu d'utilitzar el projecte `ptftpd`, un client/servidor TFTP de codi lliure escrit en Python que podeu instal·lar al vostre ordinador utilitzant la comanda `pip3 install ptftpd` un cop hagueu creat i activat l'*environment* de Python[1]. Un cop instal·lat el servidor TFTP, podeu executar-lo utilitzant la comanda '`ptftpd -D -p 6969 -r lo ./`', on el paràmetre '`-D`' activa el mode *debug*, el paràmetre '`-p`' indica el port UDP en el qual escoltarà el servidor, el paràmetre '`lo`' indica la interfície de xarxa en la qual escoltarà el servidor (*loopback*), i el paràmetre '`./`' indica el directori que conté els fitxers que es podran accedir de manera remota.

A més a més, durant el procés de desenvolupament del vostre client TFTP es recomana utilitzar Wireshark per tal de visualitzar els paquets que s'envien i es reben entre el client i el servidor TFTP, i poder verificar que segueixen el format i la seqüència adequats.

2. Descripció del protocol TFTP

Com es descriu al document RFC 1350, el protocol TFTP utilitza un model client/servidor, de manera que permet a un client llegir o escriure un fitxer del/al servidor, però el protocol no suporta ni el llistat d'arxius del servidor ni l'autenticació del client. En aquests casos es recomana la utilització del protocol FTP, definit al RFC 959.

Per tal de realitzar les comunicacions entre client i servidor, el protocol TFTP utilitza la pila IP/UDP, on per defecte el servidor escolta pel port 69 i el client utilitza un número de port aleatori (superior al 1024). Recordeu que els ports inferiors al 1024 es troben ben definits al document RFC 1340 de l'IETF. A més, en els sistemes operatius Unix aquests ports requereixen

[1] Recordeu que per crear un environment de Python heu d'executar la comanda '`python3 -m venv .`' des del directori de treball. A continuació heu d'activar l'*environment* amb la comanda '`source ./bin/activate`'.

permisos d'administrador (*root*) per executar-se, motiu pel qual heu d'utilitzar un altre port (per exemple, el 6969 tal com s'ha descrit anteriorment).

En funció del tipus de dades a transmetre, el protocol TFTP implementa tres modes de transmissió: `netascii`, `octet` i `mail`. En els tres casos la transmissió de la informació es fa amb unitats de 8 bits (1 byte), però canvia la interpretació que fan el client i servidor de les dades rebudes. En aquesta pràctica NOMÉS heu d'implementar el suport pel format OCTET, on cada caràcter es representa amb un bloc de 8 bits (1 byte) en format binari. Així doncs, la vostra implementació del protocol TFTP ha de permetre la lectura i escriptura de fitxers del servidor utilitzant el format OCTET (binari).

Per tal de realitzar la transferència el protocol TFTP defineix 5 tipus de paquets: RRQ (Read ReQuest), WRQ (Write ReQuest), DATA, ACK (Acknowledgement) i ERROR. Tal com es mostra a la Figura 1 els paquets TFTP es distingeixen pel camp `Opcode`, que sempre és el primer de la capçalera i ocupa 2 bytes. A continuació, en funció del tipus de paquet, trobem diferents camps. En el cas dels paquets RRQ i WRQ trobem els camps `Filename` i `Mode` que són una cadena de caràcters (*string*)[2] i contenen el nom del fitxer i el mode de transferència, respectivament. En canvi, en el cas dels paquets DATA i ACK el següent camp és el `Block Number`, que també ocupa 2 bytes i conté el número de seqüència per garantir que els paquets de dades i confirmació arriben en ordre. Finalment, el paquet DATA conté les dades del fitxer (fins a 512 bytes), i el paquet ERROR conté el número i la cadena de caràcters del missatge d'error.

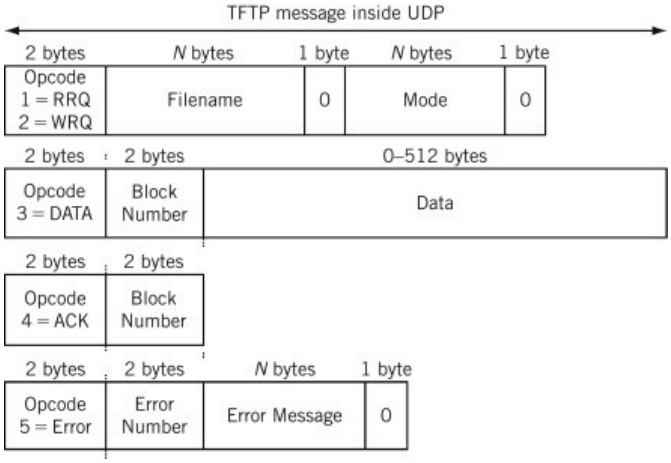

Figura 1: Format dels diferents paquets TFTP.

A continuació es descriu el procés de lectura i d'escriure d'un fitxer del client al servidor TFTP, així com la gestió que cal dur a terme en cas que es produeixin errors en la comunicació entre el client i el servidor. A més, a la Figura 2 podeu trobar un exemple del procés de lectura d'un fitxer del servidor TFTP per part del client.

[2]Recordeu que en el llenguatge de programació C les cadenes de caràcters acaben amb el caràcter 0 (NULL).

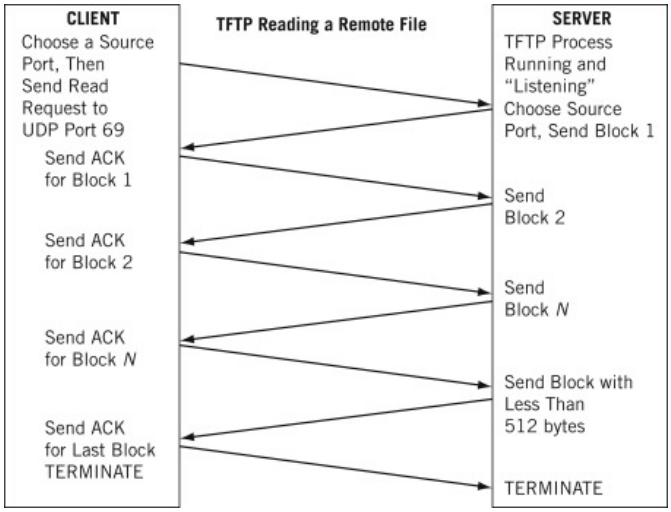

Figura 2: Sequencia d'intercanvi de paquets entre el client i el servidor TFTP.

2.1 Lectura d'un fitxer del servidor TFTP

El primer pas per realitzar la transferència d'un fitxer des del servidor TFTP, és que el client TFTP enviï un missatge RRQ al servidor TFTP amb el nom del fitxer que vol llegir.

Un cop rebut, el servidor TFTP valida que el fitxer demanat existeix en el directori local i que hi tingui permisos de lectura. En cas que el fitxer no existeixi o que no tingui permisos de lectura, el servidor TFTP respon amb un missatge d'error (OPCODE = 5) indicant el codi i el missatge d'error (per exemple, `ERROR CODE = 1, ERROR MESSAGE = "File not found."`).

En cas que el fitxer sí que existeixi i hi tingui permisos de lectura, el servidor TFTP respon amb un missatge de dades (`DATA, OPCODE = 3`) que inclou el número de bloc (`BLOCK = 1`) i el primer bloc de dades del fitxer.

Tot seguit, quan el client rep el paquet del servidor TFTP amb les dades del primer bloc, en confirma la recepció enviant un missatge de confirmació (`ACK, OPCODE = 4`) que inclou el número de bloc (`BLOCK NUMBER = 1`) que ha rebut.

Si no es produeix cap error, el procés d'enviament de paquets de dades i de confirmació es repeteix de manera iterativa entre client i servidor fins que el servidor TFTP envia el darrer bloc de dades. En aquest procés cal tenir tres punts:

- Per cada intercanvi de paquets de dades (`DATA, OPCODE = 3`) i confirmació (`ACK, OPCODE = 4`) tant el client com el servidor TFTP validen que el número de bloc correspongui amb l'esperat. En cas que no sigui l'esperat es produeix un error i finalitza la transferència.
- La mida de les dades d'un paquet de dades (`DATA, OPCODE = 3`) és sempre de 512 bytes (sense tenir en compte la capçalera), de manera que la detecció de fi de transferència és

que la mida de les dades rebudes en un paquet sigui menor de 512 bytes.

2.2 Escriptura d'un fitxer al servidor TFTP

En el cas de voler escriure un fitxer local al servidor TFTP, el primer pas que fa el client TFTP és verificar que el fitxer existeix en el directori local i que el pot llegir correctament (permís de lectura) i, a continuació, envia un paquet de petició d'escriptura (`WRQ, OPCODE = 3`) al servidor TFTP amb el nom del fitxer i el mode de transferència.

En cas que tot estigui correcte, el servidor TFTP envia un paquet de confirmació (`ACK, OPCODE = 4`) al client per indicar que ha rebut la petició correctament i pot iniciar la transferència de les dades del fitxer.

A continuació, el client TFTP llegeix el primer bloc de dades del fitxer (fins a 512 bytes de dades), prepara el paquet amb les capçaleres (`DATA, OPCODE = 3` i BLOCK NUMBER = 1) i les dades corresponents (`PAYLOAD`), i l'envia cap al servidor.

Si les dades es reben correctament, el servidor TFTP contesta amb un paquet de confirmació (ACK = 4) i el número de seqüència del paquet que ha enviat el client. Si és correcte el client pot enviar el següent paquet amb les dades. El procés es repeteix de manera iterativa fins que el client TFTP envia el darrer paquet de dades, que conté menys de 512 bytes de dades, i el servidor en confirma la recepció.

En cas que hi hagi hagut algun error en la transmissió el servidor contesta amb un paquet d'error (ERROR = 5) i el codi de l'error. En aquest cas tant client com servidor finalitzen la transferència del fitxer i informen l'usuari de l'error que s'ha produït.

2.3 Gestió d'errors en la comunicació

Com és evident, durant la comunicació entre el client i el servidor TFTP es poden produir diferents tipus d'error causats per la xarxa. El tipus d'errors més comuns i la forma de gestionar-los són els que es descriuen a continuació:

- **Timeout**: Si el client o el servidor no contesten a una petició en un període de temps determinat (per exemple, un segon) cal tornar a realitzar la petició (reenviar l'últim paquet) fins a 3 vegades. En cas que no respongui cal informar a l'usuari i finalitzar l'execució del programa.
- **Ordre**: Si el client o el servidor envien un bloc de dades no ordenat s'assumeix que hi ha hagut un error en la transmissió. En aquest cas cal informar a l'usuari de l'error i finalitzar l'execució del programa.
- **Fitxer**: Si el client o el servidor no poden llegir el fitxer sol·licitat, sigui perquè no existeix o perquè no té permisos de lectura. En aquest cas s'envia un missatge d'error per informar a l'usuari de l'error que s'ha produït i finalitza l'execució del programa.

- **Tipus**: Si el client o el servidor envien un paquet que no correspon a la seqüència s'assumeix que hi ha hagut un error en la transmissió. En aquest cas cal informar a l'usuari de l'error i finalitzar l'execució del programa.

3. Guia d'implementació de TFTP

Per tal de realitzar la implementació del client TFTP us caldrà utilitzar les llibreries `socket`, `struct` i `argparse`. Totes aquestes llibreries ja venen incloses amb l'entorn Python o bé ja les veu instal·lar a la primera pràctica.

En concret, per tal de realitzar la implementació del client TFTP haureu d'utilitzar les següents funcions:

- `socket.socket()` per crear un *socket* UDP i enviar paquets TFTP a través de la xarxa.
- `sendto()` i `recvfrom()` per tal de realitzar l'enviament i la recepció de dades a través del socket UDP.
- `timeout` per posar un temps màxim de bloqueig per llegir dades del socket UDP. En cas de no rebre dades la funció `recvfrom` retorna i permet gestionar la condició d'error (retransmissió).
- `struct.pack()` i `struct.unpack()` per tal de convertir un conjunt de variables a un vector de bytes, i a l'inrevés, d'un vector de bytes a un conjunt de variables.
- `bytes()` per tal de convertir un cadena de caràcters a un vector de bytes, i `decode()` per tal de convertir un vector de bytes a una cadena de caràcters.
- `open()` i `close()` per tal d'obrir i tancar un fitxer local abans de llegir-ne o modificar-ne el contingut.
- `read()` i `write()` per tal llegir o escriure un conjunt de dades d'un fitxer prèviament obert. Cal tancar-lo al final per assegurar que les dades es guarden a disc.
- `add_argument()` i `parse_args()` per tal d'afegir un argument a la llista de paràmetres del programa i parsejar la llista de paràmetres del programa, respectivament.

Finalment, a continuació teniu un exemple d'execució del client TFTP que llegeix el fitxer `data.txt` del servidor TFTP, ambdós executant-se a la mateixa màquina. Com es mostra a la Figura 3, el primer pas és executar el servidor TFTP pel port `6969` i la interfície `lo`. Com es pot veure, el servidor TFTP comença a executar-se i rep una petició del client TFTP per llegir el fitxer `data.txt` del *host* `127.0.0.1` (localhost). El procés finalitza correctament després de transmetre 5 paquets de dades i una mida total de 2225 bytes.

A continuació, a la Figura 3 es mostra l'execució del client TFTP. Com es pot veure el client llegeix els 5 blocs de dades i els escriu a disc abans de finalitzar.

```
(practica2) pere@neftis:~/practica2$ ./bin/ptftpd -D -p 6969 -r lo ./
INFO(tftpd): Serving TFTP requests on lo/127.0.0.1:6969 in /home/pere/practica2
INFO(tftpd): Serving file data.txt to host 127.0.0.1...
DEBUG(tftpd):   >  DATA: 5 data packet(s) sent.
DEBUG(tftpd):   <   ACK: Transfer complete, 2225 byte(s).
INFO(tftpd): Transfer of file data.txt completed.
```

Figura 3: Caption

```
(practica2) pere@neftis:~/practica2$ python3 tftp_client.py -m rx -p 6969
Starting TFTP transfer to IP=127.0.0.1 and PORT=6969 to get FILE=data.txt
Received block #1
Write data to file
Received block #2
Write data to file
Received block #3
Write data to file
Received block #4
Write data to file
Received block #5
Write data to file
Finished!
(practica2) pere@neftis:~/practica2$
```

Figura 4: Caption

Finalment, a la Figura 5 es mostra l'intercanvi de paquets entre el client i el servidor TFTP capturat amb Wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TFTP	59	Read Request, File: data.txt, Transfer type: octet
2	0.014398472	127.0.0.1	127.0.0.1	TFTP	558	Data Packet, Block: 1
3	0.014541145	127.0.0.1	127.0.0.1	UDP	46	48690 → 6969 Len=4
4	0.014655223	127.0.0.1	127.0.0.1	TFTP	558	Data Packet, Block: 2
5	0.014722845	127.0.0.1	127.0.0.1	UDP	46	48690 → 6969 Len=4
6	0.014826885	127.0.0.1	127.0.0.1	TFTP	558	Data Packet, Block: 3
7	0.014893925	127.0.0.1	127.0.0.1	UDP	46	48690 → 6969 Len=4
8	0.014973129	127.0.0.1	127.0.0.1	TFTP	558	Data Packet, Block: 4
9	0.015047247	127.0.0.1	127.0.0.1	UDP	46	48690 → 6969 Len=4
10	0.015133729	127.0.0.1	127.0.0.1	TFTP	223	Data Packet, Block: 5 (last)
11	0.015209141	127.0.0.1	127.0.0.1	UDP	46	48690 → 6969 Len=4
12	0.015369601	127.0.0.1	127.0.0.1	UDP	42	6969 → 48690 Len=0
13	0.015376354	127.0.0.1	127.0.0.1	ICMP	70	Destination unreachable (Port unreachable)

Figura 5: Captura de l'intercanvi entre el client i el servidor TFTP amb Wireshark.

4. Qüestions d'implementació

Respon amb les teves paraules les següents qüestions sobre el funcionament del protocol TFTP:

1. Quina és la mida del camp `Opcode` la capçalera TFTP, quins són els tipus de codis d'operació del protocol TFTP i quins valors tenen al camp `Opcode`?

2. Quina és la mida del camp `ErrorCode`, quins són els tipus de codis d'error del protocol TFTP i quins valors tenen al camp `ErrorCode`?

3. Com s'indica la fi d'una cadena de caràcters de mida variable en la capçalera TFTP?
4. Quin és el protocol de la capa de transport que utilitza el protocol TFTP i en quin port escolta per defecte el servidor?
5. Quina és la condició que permet a un client o servidor TFTP detectar la fi de la transferència d'un fitxer? Mostra una captura de Wireshark on s'observi.

5. Entrega i valoració de la pràctica

L'entrega de la pràctica s'haurà de fer per grups (una única entrega per grup) a través del Campus Virtual abans del **1 de novembre de 2023 a les 23:59h**. L'entrega de la pràctica consistirà en un únic fitxer en format `zip` que ha de contenir els següents elements:

- Fitxer `tftp_client.py` amb la implementació que heu realitzat a partir de la funcionalitat descrita en els apartats anteriors d'aquest enunciat.
- Fitxer `memoria.pdf` amb respostes a les preguntes, explicació i exemples d'execució del programa, captures de tràfic amb Wireshark, i explicació dels problemes que s'han trobat en el desenvolupament.

La valoració de la pràctica es realitzarà sobre 10 punts i es realitzarà tenint en compte els criteris que es detallen a continuació:

- (25%) Nivell de detall de la resposta a les preguntes formulades en aquest enunciat (incloent-hi captures de Wireshark que justifiquin la resposta).
- (50%) Qualitat del codi (estructura del codi, separació de la funcionalitat en mètodes, ús d'estructures de dades, comentaris per facilitar la comprensió, etc.).
- (25%) Qualitat de la memòria (portada, índex, explicació del funcionament, exemples de funcionament, conclusions, bibliografia consultada, etc.).

Finalment, de cara l'avaluació de la pràctica tingueu en compte aquests aspectes:

- Es durà a terme una prova de validació individual binària (aprovat/suspès) dels continguts de la pràctica que determinen la nota final el dia de l'examen parcial.
- Les entregues fora del termini establert sense justificació (mèdica, judicial, laboral, etc.) tenen una penalització de 0.5 punts per dia de demora fins a un màxim de 10 dies.
- Qualsevol indici de còpia o plagi (de companys, de fonts no citades, o d'eines com ChatGPT) comportarà que l'activitat s'avaluï amb un zero (0).
- El comportament reiterat de còpia o plagi comportarà que l'assignatura s'avaluï amb un zero (0) i l'estudiant sigui reportat a la direcció de la Universitat.