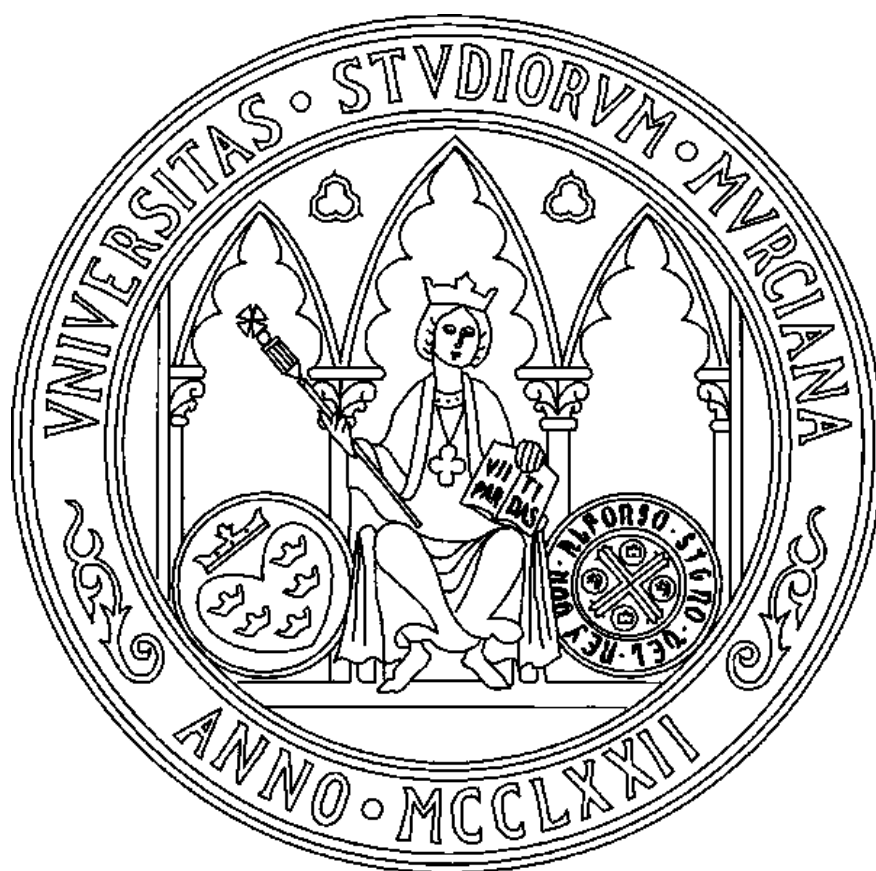


DISEÑO DE LA PRÁCTICA DE REDES DE COMUNICACIONES 2024/2025



Autores:

-Sergio Parra Martínez G1.2

-Carlos Pérez Pérez 23335807F G1.2

Profesor: Jose Rubén Titos Gil

Índice

1.Introducción.....	2
2.Formato de los mensajes del protocolo de comunicación con el Directorio.....	3
3.Formato de los mensajes del protocolo de transferencia de ficheros.....	6
4.Autómatas del protocolo.....	10
5.Ejemplo de intercambio de mensajes.....	11

1.Introducción.

En este documento **se especifica el diseño de dos protocolos**, uno que funciona para la comunicación con el directorio y otro para la transferencia de ficheros entre peers. Para ello se verá el diseño de los mensajes para dichos protocolos y los autómatas para los mismos.

En un principio íbamos a implementar 2 mejoras:

- Serve puerto efímero
- Filelist ampliado con servidores

Desafortunadamente por complicaciones y falta de tiempo solo hemos podido completar la entrega obligatoria, y por tanto estas mejoras no están hechas. Debido a esto hemos quitado de los autómatas todas las funciones relacionadas con ellos.

Además, ha sido necesario eliminar algún mensaje y añadir nuevos. El proyecto , al final , cuenta con **un send y un receive por cada instrucción implementada**. Ha sido **necesario** además implementar un nuevo mensaje **solicitud_tam** y **respuesta_tam**, debido a que en el download necesitábamos conocer el tamaño del archivo solicitado para dividirlo entre los servidores y que determinase el tamaño de chunk. Estos mensajes se han añadido a los autómatas del modelo peer to peer, entre la etapa de mandar el hash y la etapa de pedir el chunk.

Por otro lado cabe destacar que **download** cuenta con el **campo tamaño en un short**, es decir, solo se pueden descargar archivos que no sobrepasen los 32767 bits de tamaño. Esto queda demostrado en el video donde si se descarga **archivo_33000** falla la descarga pero si se descarga **archivo_32000** se completa la descarga.

2.Formato de los mensajes del protocolo de comunicación con el Directorio.

Para definir el protocolo con el cual nos comunicaremos con el Directorio, vamos a emplear mensajes textuales del tipo “**campo:valor**”. El valor que tome el campo “operation” indicará el mensaje y su formato, por lo que sabremos los campos que vienen a continuación.

Tipos y descripción de los mensajes.

Mensaje: **ping**

Sentido de la comunicación: Cliente -> Directorio

Descripción: Este mensaje lo envía el cliente de NanoFiles al directorio para ver si está activo.

Ejemplo:

```
operation: ping\nprotocolID: MORTADELA\n\n
```

Mensaje: **welcome**

Sentido de la comunicación : Directorio -> Cliente

Descripción: Este mensaje lo envía el directorio al cliente de NanoFiles para decir que está activo.

Ejemplo:

```
operation: welcome\n\n
```

Mensaje: **denied**

Sentido de la comunicación : Directorio -> Cliente

Descripción: Este mensaje lo envía el directorio al cliente de NanoFiles para decir que no está activo.

Ejemplo:

```
operation: denied\n\n
```

Mensaje: **filelist**

Sentido de la comunicación: Cliente→ Directorio

Descripción: Este mensaje lo envía el cliente de NanoFiles al Directorio para solicitar la lista de ficheros posibles para descargar.

Ejemplo:

```
operation: filelist\n
```

```
\n
```

Mensaje: **sendfilelist**

Sentido de la comunicación: Directorio→ Cliente

Descripción: Este mensaje envía al cliente la lista de ficheros disponibles, con su size, y hash.

Ejemplo:

```
operation: sendfilelist\n
```

```
filename: document.pdf\n
```

```
size: 12345\n
```

```
hash: abc123def\n
```

```
...
```

```
\n
```

Mensaje: **success**

Sentido de la comunicación: Directorio-> Cliente

Descripción: Este mensaje lo envía el directorio al cliente de NanoFiles para confirmar que lanza un servidor de ficheros que escucha conexiones en el puerto 10.000.

Ejemplo:

```
operation: success\n
```

```
\n
```

Mensaje: **serve**

Sentido de la comunicación: Cliente -> Directorio

Descripción: Este mensaje lo envía el cliente de NanoFiles al directorio para que lance un servidor de ficheros que escucha conexiones en el puerto 10.000.

Ejemplo:

```
operation: serve\n
```

```
\n
```

Mensaje: **sendserve**

Sentido de la comunicación: Directorio-> cliente

Descripción: Este mensaje lo envía el directorio al cliente en respuesta al serve.

Ejemplo:

operation: sendserve\n

\n

Mensaje: **download**

Sentido de la comunicación: Cliente→ Directorio

Descripción: Este mensaje envía al directorio una subcadena del nombre del archivo y si no imprime un mensaje de error “La lista de archivos está vacía”.

Ejemplo:

operation: download\n

filename: substring\n

\n

Mensaje: **send_download**

Sentido de la comunicación: Directorio→ Cliente

Descripción: Este mensaje envía al cliente la lista de servidores, con sus respectivas ip y número de puerto ,que contienen el fichero.

Ejemplo:

operation: send_download\n

port: 10000\n

server: 192.168.101.1\n

\n

3.Formato de los mensajes del protocolo de transferencia de ficheros.

Formatos de los mensajes:

-Control:

<i>Opcode</i>
<i>1 byte</i>

-Operación:

<i>Opcode</i>	<i>Parámetro 1</i>	<i>Parámetro 2</i>
<i>1 byte</i>	<i>8 bytes</i>	<i>8 bytes</i>

-Tamaño Variable TLV:

<i>Opcode</i>	<i>Longitud</i>	<i>Valor</i>
<i>1 byte</i>	<i>0/1/2/4/8 bytes</i>	<i>n bytes</i>

Para definir el protocolo de comunicación con un servidor de ficheros, **vamos a utilizar mensajes binarios multiformatos**. El valor que tome el campo “opcode” (código de operación) indicará el tipo de mensaje y por tanto cuál es su formato, es decir, qué campos vienen a continuación.

-Tipos y descripción de los mensajes

Mensaje: **invalid_code** (opcode = 0)

Formato: Control

Sentido de la comunicación: Servidor de ficheros → Cliente

Descripción: Este mensaje lo envía el par servidor de ficheros al par cliente (receptor) de fichero para indicar que no es un opcode valido.

Ejemplo:

<i>Opcode (1 byte)</i>
<i>0x0</i>

Mensaje: **file_not_found** (opcode = 1)

Formato: Control

Sentido de la comunicación: Servidor de ficheros → Cliente

Descripción: Este mensaje lo envía el par servidor de ficheros al par cliente (receptor) de fichero para indicar que no es posible encontrar el fichero con la información proporcionada en el mensaje de petición de descarga.

Ejemplo:

<i>Opcode (1 byte)</i>
<i>0x1</i>

Mensaje: **more_than_one** (opcode = 9)

Formato: Control

Sentido de la comunicación: Servidor de ficheros → Cliente

Descripción: Este mensaje lo envía el servidor de ficheros al par cliente (receptor) de fichero para indicar que no es posible descargar un fichero porque existen más de un archivo con ese nombre.

Ejemplo:

<i>Opcode (1 byte)</i>
<i>0x9</i>

Mensaje: **download** (opcode = 2) //Equivalente a get_hash

Formato: TLV

Sentido de la comunicación: Cliente → Servidor de ficheros.

Descripción: El par cliente manda la petición para recibir el hash del fichero buscado.

Ejemplo:

<i>Opcode (1 byte)</i>	<i>Longitud(2 bytes)</i>	<i>Valor (n bytes)</i>
<i>0x2</i>	<i>longitud_hash</i>	<i>ABC2312312</i>

Mensaje: **hash** (opcode = 3)

Formato: TLV

Sentido de la comunicación: Servidor de ficheros → Cliente

Descripción: El par del servidor manda el hash al cliente y este lo recibe.

Ejemplo:

<i>Opcode (1 byte)</i>	<i>Longitud(2 byte)</i>	<i>Valor (n bytes)</i>
<i>0x3</i>	<i>0x00 0x04</i>	<i>AB CD EF 12 AB CD EF 12</i>

Mensaje: **get_chunk** (opcode = 4)

Formato: Operación

Sentido de la comunicación: Cliente → Servidor de ficheros

Descripción: El cliente le pide al servidor el chunk.

Ejemplo:

<i>Opcode (1 byte)</i>	<i>Parametro1(8 bytes)</i>	<i>Parametro2(4 bytes)</i>
<i>0x4</i>	<i>desplazamiento</i>	<i>tam chunk</i>

Mensaje: **chunk** (opcode = 5)

Formato: TLV

Sentido de la comunicación: Servidor de ficheros → Cliente.

Descripción: El servidor de ficheros le envía al cliente el chunk que este le ha solicitado.

Ejemplo:

<i>Opcode (1 byte)</i>	<i>Parametro1(8 bytes)</i>	<i>Parametro2(depene tam_chunk)</i>
<i>0x5</i>	<i>tam_chunk</i>	<i>datos</i>

Mensaje: **solicitud_tam** (opcode = 7)

Formato: TLV

Sentido de la comunicación: Cliente → Servidor de ficheros

Descripción: El cliente le pide al servidor el tamaño del fichero.

Ejemplo:

<i>Opcode (1 byte)</i>	<i>Parametro1(8 bytes)</i>	<i>Parametro2(depene tam_nombre)</i>
<i>0x7</i>	<i>tam_nombre</i>	<i>nombre</i>

Mensaje: **respuesta_tam** (opcode = 8)

Formato: TLV

Sentido de la comunicación: Servidor de ficheros → Cliente.

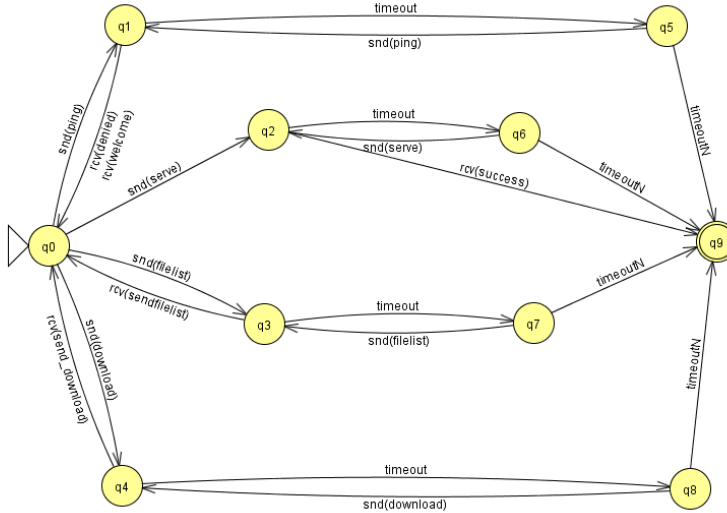
Descripción: El servidor de ficheros le envía al cliente el tamaño del archivo que este le ha solicitado.

Ejemplo:

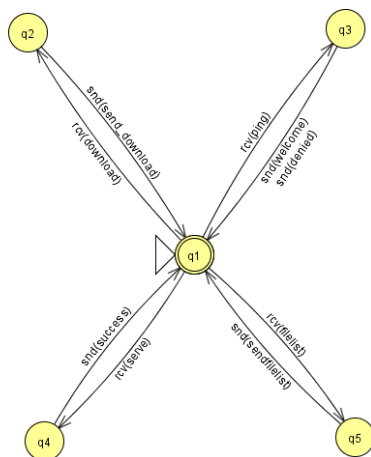
<i>Opcode (1 byte)</i>	<i>Parametro1(8 bytes)</i>	<i>Parametro2(longitud_fichero)</i>
<i>0x5</i>	<i>longitud_fichero</i>	<i>datos</i>

4. Autómatas del protocolo.

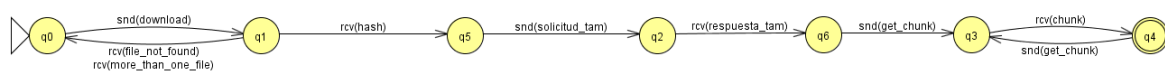
-Autómata rol cliente de directorio:



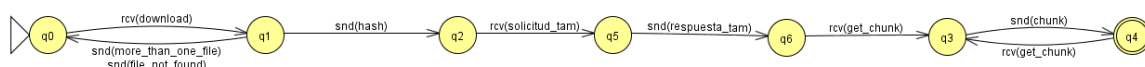
-Autómata rol servidor de directorio:



-Autómata cliente de ficheros:



-Autómata servidor de ficheros:



operation:serve filename:x size:x hash:x

1 0.000000	127.0.0.1	127.0.0.1	UDP	33 49671 → 49670 Len=1
2 0.910984	127.0.0.1	127.0.0.1	UDP	851 62355 → 6868 Len=819
3 0.912766	127.0.0.1	127.0.0.1	UDP	53 6868 → 62355 Len=21
4 1.012321	127.0.0.1	127.0.0.1	UDP	33 49671 → 49670 Len=1
5 2.021091	127.0.0.1	127.0.0.1	UDP	33 49671 → 49670 Len=1
6 3.026688	127.0.0.1	127.0.0.1	UDP	33 49671 → 49670 Len=1

Frame 2: 851 bytes on wire (6808 bits), 851 bytes captured (6808 bits) on interface \Device\NPF_{...}	0000	02 00 00 00 45 00 03 4f 69 b5 00 00 80 11 00 00	...E 0 i...
Null/Loopback	0010	7f 00 00 01 7f 00 00 01 f3 93 1a d4 03 3b b7 2b	...
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	0020	6f 70 65 72 61 74 69 6f 6e 3a 73 65 72 76 65 0a	operation:serve
User Datagram Protocol, Src Port: 62355, Dst Port: 6868	0030	70 6f 72 74 3a 31 30 30 30 30 0a 66 69 6c 65 6e	port:100 00 file
Data (819 bytes)	0040	61 6d 65 3a 61 72 63 68 69 76 6f 5f 33 32 30 30	ame:arch ivo_3200
Data [...]: 6f7065726174696f6e3a73657276650a706f72743a313030300a66696c656e616d653a6172636869766f5f333	0050	30 2e 62 69 6e 0a 73 69 7a 65 3a 33 32 30 30 30	0.bin si ze:32000
[Length: 819]	0060	0a 68 61 73 68 3a 31 65 31 64 61 37 33 39 30 61	hash:1e 1da7290a
	0070	38 63 66 62 66 65 34 35 36 66 61 37 37 35 30 65	8c8f6e45 6fa7750e
	0080	34 63 34 38 66 62 30 63 33 66 31 36 63 33 0a 66	4c48fb0c 3f16c3 f
	0090	69 6c 65 6e 61 6d 65 3a 64 65 70 6f 72 74 65 2e	ilname: deporte.
	00a0	74 78 74 0a 73 69 7a 65 3a 37 36 31 0a 68 61 73	txt size:761 has
	00b0	68 3a 35 62 66 62 36 37 35 62 32 31 65 62 31 33	h:5bfb67 53db024f
	00c0	31 65 35 61 63 39 65 37 35 62 32 31 65 62 31 33	1a5a9e7 5021eb13
	00d0	62 31 63 64 36 33 35 36 37 0a 66 69 6c 65 6e	blcd6355 67 file
	00e0	61 6d 65 3a 63 69 6e 65 2e 74 78 74 0a 73 69 7a	ame:cin. txt siz
	00f0	65 3a 37 38 37 0a 68 61 73 68 3a 65 34 62 39 61	e:787 ha sh:e4b9a
	0100	63 65 31 39 34 35 64 63 30 34 64 36 34 31 32 33	ce1945dc 04d64123
	0110	31 61 36 64 36 36 65 34 32 35 63 35 33 32 32 30	1a6d6e4 25c53220
	0120	31 34 30 0a 66 69 6c 65 6e 61 6d 65 3a 74 72 61	1a0 file name:tra
	0130	6a 73 70 6f 72 74 65 2e 74 78 74 0a 73 69 7a 65	nsporte. txt size
	0140	3a 37 38 36 0a 68 61 73 68 3a 30 65 63 64 31 62	:786 has h:0ecd1b
	0150	31 38 36 64 62 31 66 31 33 65 39 32 66 35 35 39	186db1f1 3e92f559

respuesta-> sendserve

1 0.000000	127.0.0.1	127.0.0.1	UDP	33 49671 → 49670 Len=1
2 0.910984	127.0.0.1	127.0.0.1	UDP	851 62355 → 6868 Len=819
3 0.912766	127.0.0.1	127.0.0.1	UDP	53 6868 → 62355 Len=21
4 1.012321	127.0.0.1	127.0.0.1	UDP	33 49671 → 49670 Len=1
5 2.021091	127.0.0.1	127.0.0.1	UDP	33 49671 → 49670 Len=1
6 3.026688	127.0.0.1	127.0.0.1	UDP	33 49671 → 49670 Len=1

Frame 3: 53 bytes on wire (424 bits), 53 bytes captured (424 bits) on interface \Device\NPF_{...}	0000	02 00 00 00 45 00 00 31 69 b6 00 00 80 11 00 00	...E i i...
Null/Loopback	0010	7f 00 00 01 7f 00 00 01 1a d4 f3 93 00 1d ae 98	...
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	0020	6f 70 65 72 61 74 69 6f 6e 3a 73 65 6e 64 73 65	operation:sendse
User Datagram Protocol, Src Port: 6868, Dst Port: 62355	0030	72 76 65 0a 0a	rve:...
Data (21 bytes)			
Data: 6f7065726174696f6e3a73656e6473657276650a0a			
[Length: 21]			

Operación filelist:

Cliente:

```
(nanoFiles@nf-shared) filelist
* These are the files tracked by the directory at localhost
Name                               Size Hash
texto.txt                          12 dd0d47893cc06b6a72153de729f7502a9d756a7c
prueba3                             920 8fc53765d1a11143402b3ce6f47ba43a8b1b7ff2
por                                 27 364a16b58f90dedb4e5ed3014a1ca8cb3121d98b
(nanoFiles@nf-shared)
```

Servidor:

```
Directory received datagram from /127.0.0.1:57276 of size 20 bytes.
Data received: operation:filelist
```

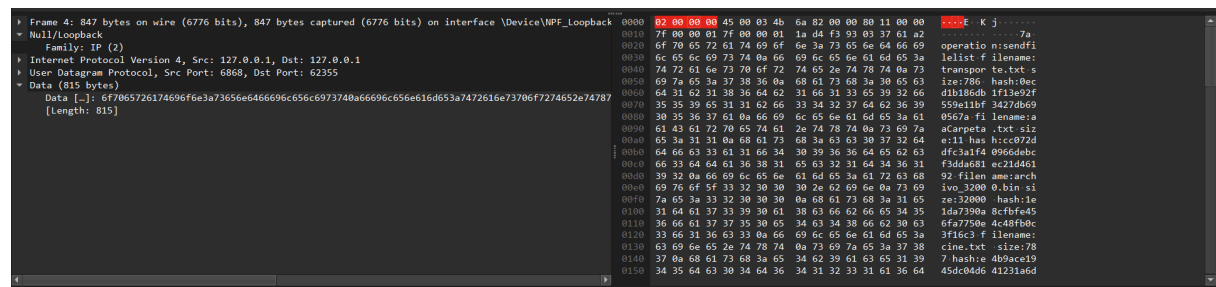
operation:filelist

1 0.000000	127.0.0.1	127.0.0.1	UDP	33 49671 → 49670 Len=1
2 1.020996	127.0.0.1	127.0.0.1	UDP	33 49671 → 49670 Len=1
3 1.146169	127.0.0.1	127.0.0.1	UDP	52 62355 → 6868 Len=20
4 1.150868	127.0.0.1	127.0.0.1	UDP	847 6868 → 62355 Len=815
5 2.029582	127.0.0.1	127.0.0.1	UDP	33 49671 → 49670 Len=1
6 3.041388	127.0.0.1	127.0.0.1	UDP	33 49671 → 49670 Len=1

Frame 3: 52 bytes on wire (416 bits), 52 bytes captured (416 bits) on interface \Device\NPF_{...}	0000	02 00 00 00 45 00 00 30 6a 81 00 00 80 11 00 00	...E 0 j...
Null/Loopback	0010	7f 00 00 01 7f 00 00 01 f3 93 1a d4 00 1c 28 54	...
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1	0020	6f 70 65 72 61 74 69 6f 6e 3a 66 69 6c 65 6c 69	operation:fileli
User Datagram Protocol, Src Port: 62355, Dst Port: 6868	0030	73 74 0a 0a	st:...
Data (20 bytes)			
Data: 6f7065726174696f6e3a66696c656c6973740a0a			
[Length: 20]			

operation:send_filelist

1 0.000000	127.0.0.1	127.0.0.1	UDP	33 49671 → 49670 Len=1
2 1.020996	127.0.0.1	127.0.0.1	UDP	33 49671 → 49670 Len=1
3 1.146169	127.0.0.1	127.0.0.1	UDP	52 62355 → 6868 Len=20
4 1.150868	127.0.0.1	127.0.0.1	UDP	847 6868 → 62355 Len=815
5 2.029582	127.0.0.1	127.0.0.1	UDP	33 49671 → 49670 Len=1
6 3.041388	127.0.0.1	127.0.0.1	UDP	33 49671 → 49670 Len=1

**Cliente:**

```
(nanoFiles@nf-shared) download pru prueba4
New client connected: /127.0.0.1:59797
Se ha completado la descarga .
(nanoFiles@nf-shared)
```

```
Directory received datagram from /127.0.0.1:57276 of size 33 bytes.  
Data received: operation:download  
filename:pru  
  
Archivo encontrado: prueba3
```

filename:transport

3	1.332481	127.0.0.1	127.0.0.1	UDP	70 62355 + 6868 Len=38	
4	1.335623	127.0.0.1	127.0.0.1	UDP	86 6868 + 62355 Len=54	
5	1.365221	127.0.0.1	127.0.0.1	TCP	56 50854 + 10000 [SWM] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM	
6	1.365202	127.0.0.1	127.0.0.1	TCP	56 10000 + 50854 [ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM	
7	1.365319	127.0.0.1	127.0.0.1	TCP	44 50854 + 10000 [ACK] Seq=1 Ack=1 Win=65280 Len=0	
8	1.378629	127.0.0.1	127.0.0.1	TCP	45 50854 + 10000 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=1 [TCP PDU reassembled in 12]	
9	1.378667	127.0.0.1	127.0.0.1	TCP	44 10000 + 50854 [ACK] Seq=1 Ack=2 Win=65280 Len=0	
10	1.379638	127.0.0.1	127.0.0.1	TCP	46 50854 + 10000 [PSH, ACK] Seq=2 Ack=1 Win=65280 Len=2 [TCP PDU reassembled in 12]	
11	1.379657	127.0.0.1	127.0.0.1	TCP	44 10000 + 50854 [ACK] Seq=1 Ack=4 Win=65280 Len=0	
12	1.379689	127.0.0.1	127.0.0.1	TCP	52 50854 + 10000 [PSH, ACK] Seq=4 Ack=1 Win=65280 Len=0	
13	1.379694	127.0.0.1	127.0.0.1	TCP	44 10000 + 50854 [ACK] Seq=1 Ack=12 Win=65280 Len=0	
14	1.379648	127.0.0.1	127.0.0.1	TCP	45 10000 + 50854 [PSH, ACK] Seq=1 Ack=12 Win=65280 Len=1 [TCP PDU reassembled in 18]	
15	1.379868	127.0.0.1	127.0.0.1	TCP	44 50854 + 10000 [ACK] Seq=12 Ack=2 Win=65280 Len=0	
16	1.379891	127.0.0.1	127.0.0.1	TCP	46 10000 + 50854 [PSH, ACK] Seq=2 Ack=12 Win=65280 Len=2 [TCP PDU reassembled in 18]	
17	1.379900	127.0.0.1	127.0.0.1	TCP	44 50854 + 10000 [ACK] Seq=12 Ack=2 Win=65280 Len=0	
18	1.379927	127.0.0.1	127.0.0.1	TCP	84 10000 + 50854 [PSH, ACK] Seq=4 Ack=12 Win=65280 Len=40	
19	1.379936	127.0.0.1	127.0.0.1	TCP	44 50854 + 10000 [ACK] Seq=12 Ack=44 Win=65280 Len=0	
20	1.379988	127.0.0.1	127.0.0.1	TCP	45 50854 + 10000 [PSH, ACK] Seq=12 Ack=44 Win=65280 Len=1 [TCP PDU reassembled in 24]	

```

# Frame 10: 60 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface \Device\NPF_{...}
# Ethernet II, Src: Intel(R) Ethernet Controller (P0-P3) 82:00:00:00:00:00, Dst: Intel(R) Ethernet Controller (P0-P3) 82:00:00:00:00:00
# Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
# User Datagram Protocol, Src Port: 62355, Dst Port: 6868
# Data (38 bytes)
0000  67 065726174696f63a646776e6cf61640a66969c656e16d653a742616ef73706f72b0da
[Length: 38]

```

server:x

```

1      4 1.356263 127.0.0.1 127.0.0.1 UDP 86 6868 → 62355 Len=54
2      5 1.365221 127.0.0.1 127.0.0.1 TCP 56 50854 → 10000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
3      6 1.365302 127.0.0.1 127.0.0.1 TCP 56 10000 → 50854 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
4      7 1.365319 127.0.0.1 127.0.0.1 TCP 44 50854 → 10000 [ACK] Seq=1 Ack=1 Win=65280 Len=0
5      8 1.378629 127.0.0.1 127.0.0.1 TCP 45 50854 → 10000 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=1 [TCP PDU reassembled in 12]
6      9 1.378667 127.0.0.1 127.0.0.1 TCP 44 10000 → 50854 [ACK] Seq=1 Ack=2 Win=65280 Len=0
7     10 1.379638 127.0.0.1 127.0.0.1 TCP 46 50854 → 10000 [PSH, ACK] Seq=2 Ack=1 Win=65280 Len=2 [TCP PDU reassembled in 12]
8     11 1.379657 127.0.0.1 127.0.0.1 TCP 44 10000 → 50854 [ACK] Seq=1 Ack=4 Win=65280 Len=0
9     12 1.379689 127.0.0.1 127.0.0.1 TCP 52 50854 → 10000 [PSH, ACK] Seq=4 Ack=1 Win=65280 Len=8
10    13 1.379694 127.0.0.1 127.0.0.1 TCP 44 10000 → 50854 [ACK] Seq=1 Ack=12 Win=65280 Len=0
11    14 1.379848 127.0.0.1 127.0.0.1 TCP 45 10000 → 50854 [PSH, ACK] Seq=1 Ack=12 Win=65280 Len=1 [TCP PDU reassembled in 18]
12    15 1.379868 127.0.0.1 127.0.0.1 TCP 44 50854 → 10000 [ACK] Seq=12 Ack=1 Win=65280 Len=0
13    16 1.379891 127.0.0.1 127.0.0.1 TCP 45 10000 → 50854 [PSH, ACK] Seq=12 Ack=12 Win=65280 Len=2 [TCP PDU reassembled in 18]
14    17 1.379900 127.0.0.1 127.0.0.1 TCP 44 50854 → 10000 [ACK] Seq=12 Ack=4 Win=65280 Len=0
15    18 1.379927 127.0.0.1 127.0.0.1 TCP 84 10000 → 50854 [PSH, ACK] Seq=4 Ack=12 Win=65280 Len=40
16    19 1.379936 127.0.0.1 127.0.0.1 TCP 44 50854 → 10000 [ACK] Seq=12 Ack=4 Win=65280 Len=0
20    20 1.379988 127.0.0.1 127.0.0.1 TCP 44 50854 → 10000 [ACK] Seq=12 Ack=4 Win=65280 Len=0
Frame 4: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface UDevice\NPF_{...}
Null/Loopback
Family: IP (2)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
User Datagram Protocol, Src Port: 6868, Dst Port: 62355
Data (54 bytes)
Data: 6f706572617469666e3a373656645f646f776e6cf6f640a706f72743a31303030308a73655727665723a2f3132327e3
[Length: 52]
0000  02 00 00 00 45 00 00 52  6b b5 00 00 00 11 00 00  .....E r k
0010  7f 00 00 01 7f 00 00 01  1a d4 f3 93 00 3e 2a 8a  .....>8
0020  6f 70 65 72 61 74 69 6f  6e 3a 73 65 6e 64 5f 64  operation send d
0030  5f 77 6e 6f 6f 64 0a 70  6f 72 74 3a 31 30 30 30  ..download port:108
0040  30 30 8a 73 65 72 6f 65  72 7a 2f 31 32 32 7e 3e  80=serve r:/127.0
0050  2e 30 2e 31 30 0a 00 00  ..0.1..

```

13

```

42 1.381133 127.0.0.1 127.0.0.1 TCP 830 10000 → 50854 [PSH, ACK] Seq=53 Ack=36 Win=65280 Len=786
43 1.381147 127.0.0.1 127.0.0.1 TCP 44 50854 → 10000 [ACK] Seq=36 Ack=839 Win=64512 Len=0

Frame 42: 830 bytes on wire (6640 bits), 830 bytes captured (6640 bits) on interface \Device\NPF_{...}
Null/Loopback
Family: IP (2)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 10000, Dst Port: 50854, Seq: 53, Ack: 36, Len: 786
[3 Reassembled TCP Segments (789 bytes): #30(1), #40(2), #42(786)]
Data (789 bytes)
0000 02 00 00 00 45 00 03 3a 6b db 40 00 80 06 00 00 ... E : k 0 ...
0010 7f 00 00 01 7f 00 00 01 27 10 c6 a6 88 7b 9a 6f ... ..... { o
0020 0c 4b f2 9e 50 18 00 ff 7c 94 00 00 4c 61 73 20 ... K . P ... Las
0030 63 69 75 64 61 64 65 73 20 6d 6f 64 65 72 6e 61 ... ciudades moderna
0040 73 20 65 6e 66 72 65 6e 74 61 6e 20 67 72 61 6e ... s enfren tan gran
0050 64 65 73 20 64 65 73 61 66 c3 ad 6f 73 20 65 6e ... des desa f os en
0060 20 6d 6f 76 69 6c 69 64 61 64 20 75 72 62 61 6e ... moviliad urban
0070 61 2e 20 45 6c 20 74 72 61 6e 73 70 6f 72 74 65 ... a. El tr ansporte
0080 20 70 c3 ba 62 6c 69 63 6f 20 65 6e 69 63 69 65 ... p blic o eficie
0090 6e 74 65 20 65 73 20 63 6c 61 76 65 20 70 61 72 ... nte es c lave par
00a0 61 20 72 65 64 75 63 69 72 20 6c 61 20 63 6f 6e ... a reduci r la con
00b0 67 65 73 74 69 c3 b3 6e 20 79 20 6c 61 20 63 6f ... gesti n y la co
00c0 6e 74 61 6d 69 6e 61 63 69 c3 b3 6e 2e 20 4c 6f ... ntaminac i n. Lo
00d0 73 20 61 75 74 6f 62 75 73 65 73 20 65 6c c3 a9 ... s autobu ses el
00e0 63 74 72 69 63 6f 73 20 79 20 6c 6f 73 20 74 72 ... ctricos y los tr
00f0 65 6e 65 73 20 64 65 20 61 6c 74 61 20 76 65 6c ... enes de alta vel
0100 6f 63 69 64 61 64 20 73 6f 6e 20 65 6a 65 6d 70 ... ocidad s on ejemp
0110 6c 6f 73 20 64 65 20 69 6e 6e 6f 76 61 63 69 c3 ... los de i nnovaci
0120 b3 6e 20 73 6f 73 74 65 6e 69 62 6c 65 2e 20 4c ... n soste nible. L
0130 61 73 20 62 69 63 69 63 6c 65 74 61 73 20 79 20 ... as bisi c letas y
0140 6c 6f 73 20 70 61 74 69 6e 65 74 65 73 20 65 6c ... los pati netes el

```

Operación download en otro peer (mismo localhost, otra terminal):

Cliente:

```

(nanoFiles@nf-shared) download pru prueba4
Se ha completado la descarga .
(nanoFiles@nf-shared)

```

Servidor Peer(tras varias descargas):

```

(nanoFiles@nf-shared) New client connected: /127.0.0.1:59818
New client connected: /127.0.0.1:59848
New client connected: /127.0.0.1:59851
New client connected: /127.0.0.1:59861
New client connected: /127.0.0.1:59868
New client connected: /127.0.0.1:59869
New client connected: /127.0.0.1:59884
New client connected: /127.0.0.1:59888

```

Directory:

```

Archivo encontrado: prueba3
Directory received datagram from /127.0.0.1:63228 of size 33 bytes.
Data received: operation:download
filename:pru
Archivo encontrado: prueba3

```

Tratamiento errores:

-Descargar y querer guardar con el mismo nombre:

```

(nanoFiles@nf-shared) filelist
* These are the files tracked by the directory at localhost
Name Size Hash
texto.txt 12 dd0d47893cc06b6a72153de729f7502a9d756a7c
prueba3 920 8fc53765d1a11143402b3ce6f47ba43a8b1b7ff2
por 27 364a16b58f90dedb4e5ed3014a1ca8cb3121d98b
(nanoFiles@nf-shared) download pru prueba3
Ya existe un fichero con ese nombre:prueba3

```

-Descargar y que encuentre varios con la misma subcadena:

```

(nanoFiles@nf-shared) download p casa
Error 0 o mas de un archivo con ese nombre

```

-Intentar abrir dos directorios:

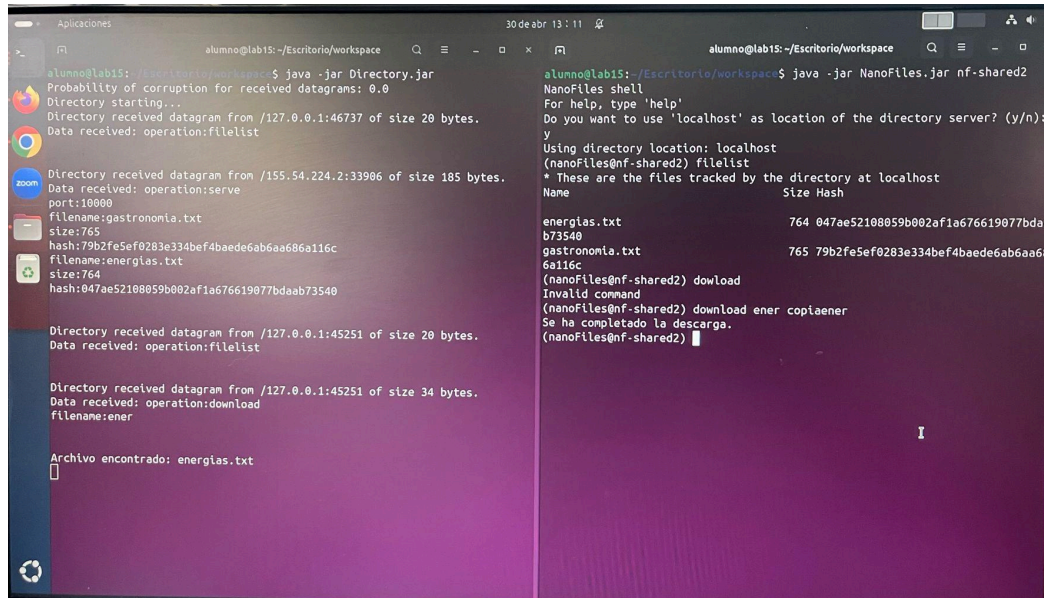
```

Probability of corruption for received datagrams: 0.0
Directory cannot create UDP socket
Most likely a Directory process is already running and listening on that port...

```


Desde pcs de los laboratorios

-Directorio y NanoFiles IP:155.24.224.1:



The image shows two terminal windows side-by-side. The left window shows the Directory application running, displaying logs for data received from various IP addresses and file operations like 'operation:filelist' and 'operation:download'. The right window shows the NanoFiles application running, displaying a shell prompt and a list of files tracked by the directory at localhost.

```
alumno@lab15: ~/Escritorio/workspace
alumno@lab15:~/Escritorio/workspace$ java -jar Directory.jar
Probability of corruption for received datagrams: 0.0
Directory starting...
Directory received datagram from /127.0.0.1:46737 of size 20 bytes.
Data received: operation:filelist

Directory received datagram from /155.54.224.2:33986 of size 185 bytes.
Data received: operation:serve
port:10000
filename:gastronomia.txt
size:765
hash:79b2fe5ef0283e334bef4baede6ab6aa686a116c
filename:energias.txt
size:764
hash:047ae52108059b002af1a676619077bdaab73540

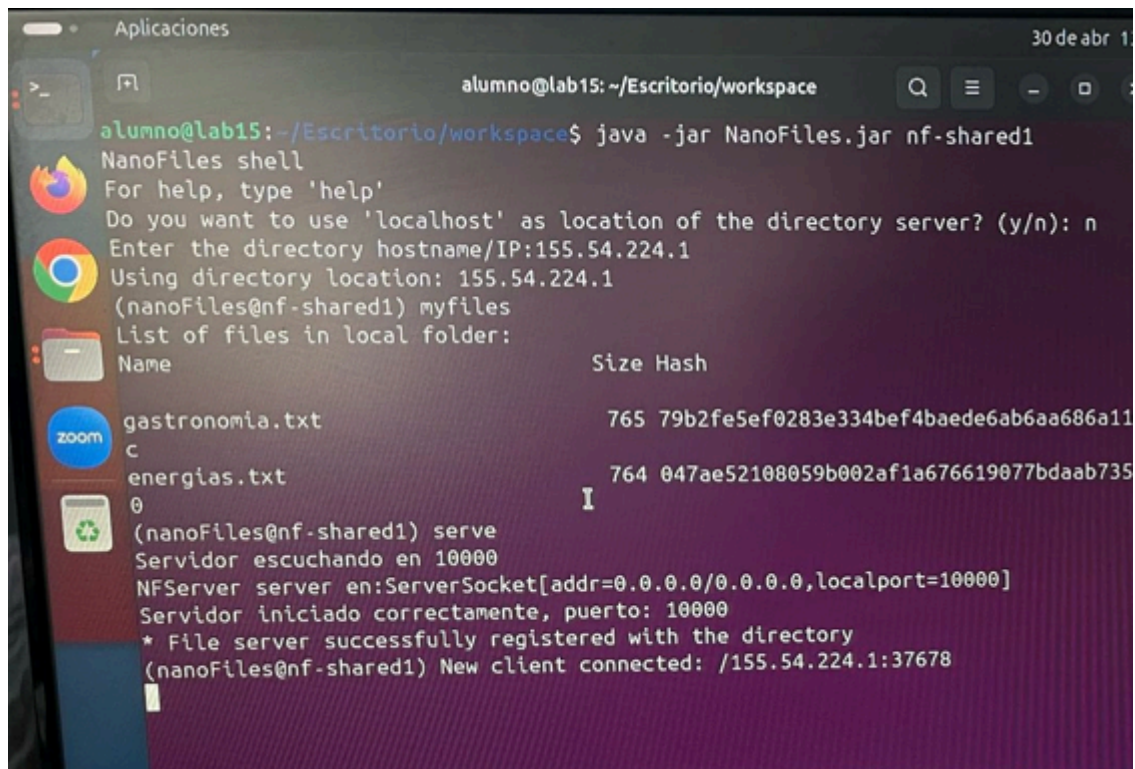
Directory received datagram from /127.0.0.1:45251 of size 20 bytes.
Data received: operation:filelist

Directory received datagram from /127.0.0.1:45251 of size 34 bytes.
Data received: operation:download
filename:ener

Archivo encontrado: energias.txt

alumno@lab15:~/Escritorio/workspace$ java -jar NanoFiles.jar nf-shared2
NanoFiles shell
For help, type 'help'
Do you want to use 'localhost' as location of the directory server? (y/n): y
Using directory location: localhost
(nanoFiles@nf-shared2) filelist
* These are the files tracked by the directory at localhost
Name                                     Size Hash
energias.txt                             764 047ae52108059b002af1a676619077bdaab73540
gastronomia.txt                          765 79b2fe5ef0283e334bef4baede6ab6aa686a116c
(nanoFiles@nf-shared2) download
Invalid command
(nanoFiles@nf-shared2) download ener copiaener
Se ha completado la descarga.
(nanoFiles@nf-shared2)
```

-NanoFiles en IP 155.24.224.2:(Necesario la conexion a la ip del Directorio)



The image shows a terminal window with the NanoFiles application running. It prompts the user to enter a directory hostname/IP, which is set to 155.54.224.1. It then displays a list of files in the local folder and shows the server status.

```
alumno@lab15:~/Escritorio/workspace$ java -jar NanoFiles.jar nf-shared1
NanoFiles shell
For help, type 'help'
Do you want to use 'localhost' as location of the directory server? (y/n): n
Enter the directory hostname/IP:155.54.224.1
Using directory location: 155.54.224.1
(nanoFiles@nf-shared1) myfiles
List of files in local folder:
Name                                     Size Hash
gastronomia.txt                          765 79b2fe5ef0283e334bef4baede6ab6aa686a116c
energias.txt                             764 047ae52108059b002af1a676619077bdaab73540
(nanoFiles@nf-shared1) serve
Servidor escuchando en 10000
NFServer server en:ServerSocket[addr=0.0.0.0/0.0.0.0,localport=10000]
Servidor iniciado correctamente, puerto: 10000
* File server successfully registered with the directory
(nanoFiles@nf-shared1) New client connected: /155.54.224.1:37678
```