Facultad de Informática

UNIVERSIDAD DE MURCIA

DOCUMENTO DE DISEÑO

PROYECTO NANOFILES

Asignatura: Redes de Computadores

Autor: Juan Rondán Ramos. DNI: 48854653P

Autor: Luis Molina Llamos. DNI: 55142089A

Profesor: Juan José Pujante Moreno

Convocatoria de Mayo 2025

Facultad de Informática

Universidad de Murcia

Índice

Contenido

Introducción.

El sistema NanoFiles está formado por un servidor de directorio (programa Directory) y un conjunto de peers o pares (programa NanoFiles), que se comunican entre sí de la siguiente forma:

- Por un lado, la comunicación entre cada peer de NanoFiles y el servidor de directorio se rige por el modelo cliente-servidor. Un peer actúa como cliente del directorio para consultar los ficheros que pueden ser descargados de otros peers, publicar los ficheros que quiere compartir con el resto de pares y obtener los servidores que comparten un determinado fichero.

- Por otro lado, el modelo de comunicación entre pares de NanoFiles es peer-to-peer (P2P).

    - Cuando un peer actúa como cliente de otro peer servidor, el cliente puede consultar los ficheros disponibles en el servidor y descargar aquellos fragmentos (chunks) de un fichero determinado que solicite.
    - De manera complementaria, un peer puede convertirse a petición del usuario en servidor de ficheros, de forma que escuche en un puerto determinado en espera de que otros peers se conecten para solicitarle fragmentos de los ficheros que está compartiendo.

Protocolos diseñados

Directorio

Formato de los mensajes y ejemplos de los mismos

Para el intercambio de mensajes y comunicación entre el directorio y un peer servidor, nos centraremos en el formato campo:valor. Este formato utiliza delimitadores basados en texto y facilita la legibilidad de los mensajes. El protocolo de comunicación entre Directorio y peer cliente se establece sobre un canal no confiable UDP. Los mensajes ASCII diseñados para este intercambio de información son los siguientes:

- Mensaje: ping
  Sentido de la comunicación: Peer cliente -> Directorio
  Descripción: El peer cliente contacta con el servidor de directorio para comprobar que está a la escucha y que utiliza un protocolo compatible con el suyo.

- Mensaje: pingok
  Sentido de la comunicación: Directorio -> Peer cliente
  Descripción: El Directorio responde al peer cliente informándole de que está a la escucha y de que usa un protocolo compatible.

- Mensaje: pingError
  Sentido de la comunicación: Directorio -> Peer cliente
  Descripción: El Directorio responde al peer cliente informándole de que ha ocurrido un error y no se ha podido establecer la conexión.

    - Ejemplo de intercambio de mensaje:

        - Caso de acierto:

          Peer -> Directorio
          operation: ping
          protocol: 48854653P-55142089A

          Directorio -> Peer
          operation: pingok

        - Caso de fallo:

          Peer -> Directorio
          operation: ping
          protocol: 48854653P-55142089A

          Directorio -> Peer
          operation: pingError

- Mensaje: serve
  Sentido de la comunicación: Peer cliente -> Directorio
  Descripción: El peer cliente lanza una solicitud para publicar su lista de ficheros compartidos, presentes en la carpeta nf-shared.

- Mensaje: serveok
  Sentido de la comunicación: Directorio -> Peer cliente
  Descripción: El Directorio responde al peer cliente indicándole que sus ficheros compartidos en la carpeta nf-shared se han publicado con éxito.

- Mensaje: serveError
  Sentido de la comunicación: Directorio -> Peer cliente
  Descripción: El Directorio responde al peer cliente indicándole que no hay ficheros compartidos en la carpeta nf-shared y que por tanto no se puede publicar nada.

    - Ejemplo de intercambio de mensaje:

        - Caso de acierto:

          Peer -> Directorio
          operation: serve
          port: 57611
          filecount: 2
          filename: prueba.txt
          size: 150
          hash: ed1f33982fb019e3ab924a635227c290dceddbd6
          filename: otro.txt
          size: 61
          hash: 331fb8c612206a30c691eda742ef5cd760b74551

          Directorio -> Peer
          operation: serveok

        - Caso de fallo:

          Peer -> Directorio
          operation: serve
          port: 57611
          filecount: 0

          Directorio -> Peer
          operation: serveError

- Mensaje: filelist
  Sentido de la comunicación: Peer cliente -> directorio
  Descripción: El peer cliente lanza una solicitud para mostrar la lista de ficheros que han sido publicados al Directorio

- Mensaje: filelistok
  Sentido de la comunicación: Directorio -> Peer cliente
  Descripción: El Directorio responde al peer mostrando la lista de ficheros que hay publicados.

- Mensaje: filelistError
  Sentido de la comunicación: Directorio -> Peer cliente
  Descripción: El Directorio responde al peer cliente indicando que ningún fichero ha sido publicado.

  - Ejemplo de intercambio de mensaje:

    - Caso de acierto:

      Peer -> Directorio
      operation: filelist

      Directorio -> Peer
      operation: filelistok
      filecount: 2
      filename: prueba.txt
      size: 150
      hash: ed1f33982fb019e3ab924a635227c290dceddbd6
      filename: otro.txt
      size: 61
      hash: 331fb8c612206a30c691eda742ef5cd760b74551

    - Caso de fallo:

      Peer -> Directorio
      operation: filelist

      Directorio -> Peer
      operation: filelistError

- Mensaje: myfiles
  Sentido de la comunicación: Local
  Descripción: El peer muestra en pantalla los ficheros que tiene localmente en la carpeta nf-shared, indicando nombre, tamaño y hash.

  - Ejemplo de intercambio de mensaje:

- Caso de tener ficheros en la carpeta nf-shared:

  Peer -> Directorio
  operation: myfiles

  Directorio -> Peer
  List of files in local folder:

Name	Size	Hash
prueba.txt	10	ed1f33982fb019e3ab924
foto.jpg	485918	25312d3c18c25f64edf70d
hola.txt	61	91eda742ef5cd760b74551

- Caso de no tener ficheros en la carpeta nf-shared:

  Peer -> Directorio
  operation: myfiles

  Directorio -> Peer
  List of files in local folder:

Name	Size	Hash

(mostramos la tabla vacía, no hay respuesta de acierto ni de error)

- Mensaje: download <subcadena fichero a buscar> <nombre archivo descargado>
  Sentido de la comunicación: Peer cliente -> Directorio
  Descripción: El peer cliente solicita descargar un fichero de los existentes en el servidor de Directorio, cuyo nombre encaje con la subcadena introducida en el primer parámetro

- Mensaje: downloadok
  Sentido de la comunicación: Directorio -> Peer cliente
  Descripción: El Directorio indica que existe un archivo con la subcadena indicada, y que la descarga ha sido realizada con éxito.

- Mensaje: downloadError
  Sentido de la comunicación: Directorio -> Peer cliente
  Descripción: El Directorio indica que no existe un fichero con la subcadena recibida o que la propia descarga ha fallado.

  - Ejemplo de intercambio de mensaje:

    - Caso de acierto:

      Peer -> Directorio

operation: download prueba.txt new_name.txt

Directorio -> Peer
operation: downloadok

- Caso de fallo:

Peer -> Directorio
operation: download text.txt nombre.txt

Directorio -> Peer
operation: downloadError

- Mensaje: help
  Sentido de la comunicación: Local
  Descripción: Muestra ayuda sobre las órdenes soportadas

  - Ejemplo de intercambio de mensaje:

    Peer -> Directorio
    operation: help

    Directorio -> Peer
    List of commands:
    quit -- quit the application
    ping -- ping directory to check protocol compatibility
    filelist -- show list of files tracked by the directory
    myfiles -- show contents of local folder (files that may be served)
    serve -- run file server and publish served files to directory
    download -- download file from all available server(s)
    help -- shows this information

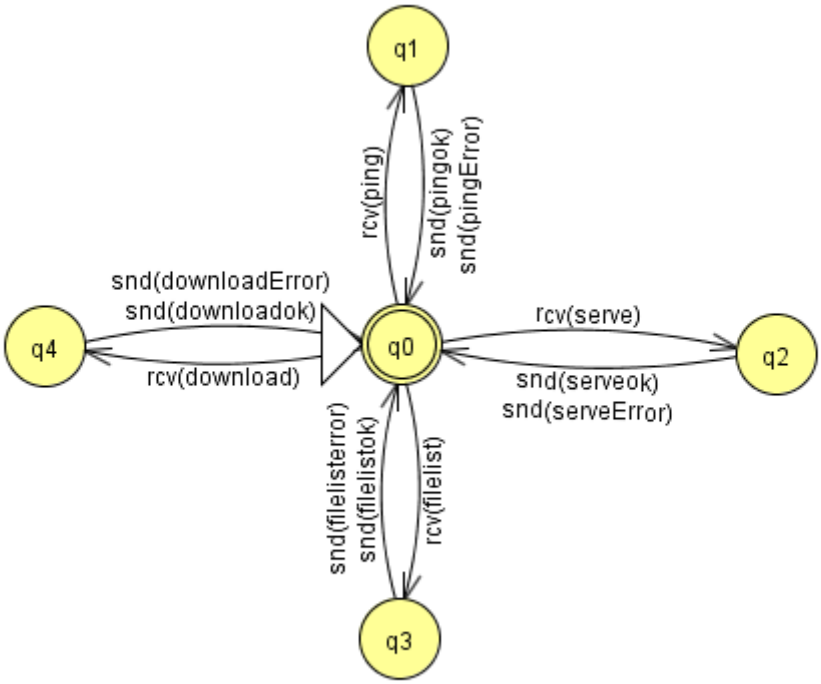
- Mensaje: quit
  Sentido de la comunicación: Local
  Descripción: Sale de la aplicación

    Peer -> Directorio
    operation: quit

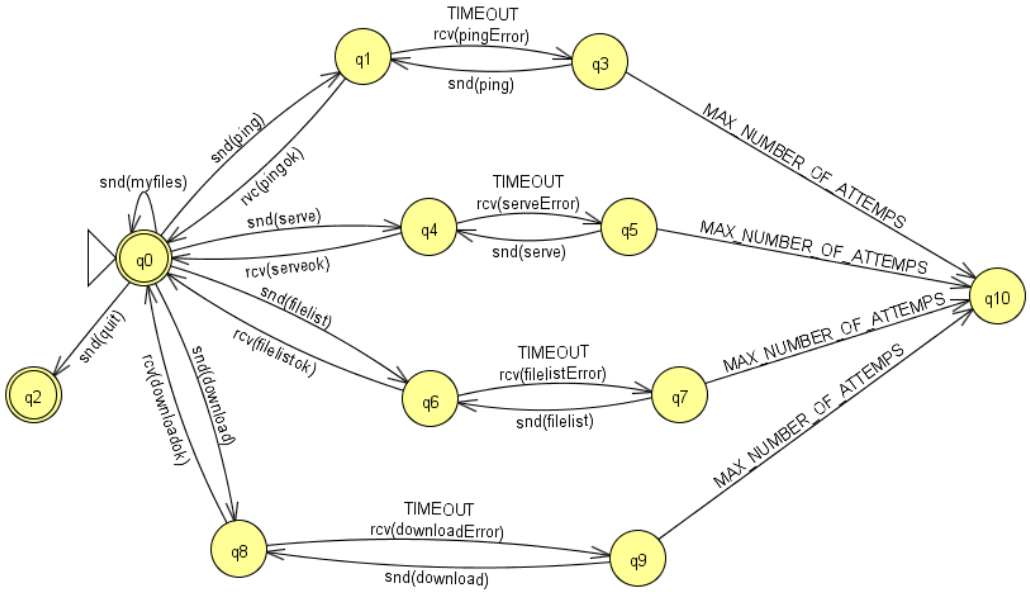
    Directorio -> Peer
    Bye.

Autómatas cliente y servidor

- Autómata Directorio:



- Autómata Peer cliente



Peer To Peer

Formato de los mensajes y ejemplos de los mismos

Para llevar a cabo la transferencia de ficheros entre peers, se han implementado mensajes binarios multiformato que son intercambiados entre un peer servidor de ficheros y un peer cliente que desea descargar los ficheros disponibles. El protocolo de comunicación entre cliente y servidor de ficheros se establece sobre un canal confiable TCP. Los mensajes binarios diseñados son los siguientes:

- Mensaje: FILE_NOT_FOUND
  Sentido de la comunicación: Peer servidor -> Peer cliente
  Descripción: El servidor de ficheros responde con este mensaje cuando no encuentra un fichero que coincida con la subcadena solicitada o cuando el número de chunk solicitado está fuera de rango respecto al tamaño del fichero.
  Opcode: 0x01
  Formato del mensaje:

Opcode
1 byte

- Mensaje: FILE_INFO_REQUEST
  Sentido de la comunicación: Peer cliente -> Peer servidor
  Descripción: El cliente solicita al servidor de ficheros información detallada (nombre, tamaño y hash) de un fichero cuyo nombre contenga la subcadena indicada. El servidor responderá con los datos del primer fichero coincidente.
  Opcode: 0x04
  Formato del mensaje:

Opcode	Subcadena
1 byte	n bytes

- Mensaje: FILE_INFO_RESPONSE
  Sentido de la comunicación: Peer servidor -> Peer cliente
  Descripción: El servidor de ficheros responde al FILE_INFO_REQUEST proporcionando el nombre, el tamaño y el hash del primer fichero cuyo nombre contiene la subcadena enviada por el cliente.
  Opcode: 0x05
  Formato del mensaje:

Opcode	Nombre fichero	Tamaño	Hash
1 byte	n bytes	8 bytes	m bytes

- Mensaje: GET_CHUNK
  Sentido de la comunicación: Peer cliente -> Peer Servidor
  Descripción: El cliente solicita a un peer servidor un fragmento (chunk) específico de un fichero determinado. A partir de la información recibida en el FILE_INFO_RESPONSE (tamaño del fichero) y el número de peers que tengan el fichero, obtenido al hacer el propio download, el cliente calcula la división en chunks. Luego envía un mensaje GET_CHUNK a cada peer, indicando el número de chunk y el tamaño que se espera recibir. Cada peer servidor recibe un GET_CHUNK único, indicando qué parte exacta del fichero debe enviar.
  Opcode: 0x02
  Formato del mensaje:

Opcode	Nombre fichero	Número de chunk	Tamaño de chunk
1 byte	n bytes	4 bytes	m bytes

- Mensaje: SEND_CHUNK
  Sentido de la comunicación: Peer servidor -> Peer cliente
  Descripción: El servidor responde enviando el fragmento solicitado en el GET_CHUNK. Al recibir el tamaño del chunk y el número de chunk, el servidor calcula localmente el offset en el fichero, lee el número de bytes solicitados y envía el fragmento binario como campo de datos en el mensaje.
  Formato del mensaje:

Opcode	Nombre fichero	Número de chunk	Datos
1 byte	n bytes	4 bytes	m bytes

- Ejemplo de intercambio de mensajes:

  - Caso de acierto:

  Cliente -> Servidor

  Mensaje: FILE_INFO_REQUEST

  Se solicita información sobre el fichero cuyo nombre contiene una subcadena dada.

  Servidor -> Cliente

  Mensaje: FILE_INFO_RESPONSE

  Responde con el nombre exacto, tamaño y hash del fichero.

  Localmente en el cliente:

  Calcula el tamaño del chunk (chunkSize).

  Divide el fichero en chunks y asigna un chunk a cada peer disponible.

  Cliente -> Servidor

  Mensaje: GET_CHUNK (uno por cada peer servidor disponible).

Solicita el chunk específico al servidor, indicándole número de chunk y tamaño.

Localmente en el servidor:

Calcula el offset, y la cantidad exacta de bytes que debe leer.

Servidor -> Cliente:

Mensaje: SEND_CHUNK (en respuesta individual a cada GET_CHUNK).

Responde enviando el fragmento binario del fichero correspondiente.

Cliente:

Ensambla los chunks recibidos y verifica la integridad final, comparando el hash calculado con el esperado.

- Caso de fallo

Cliente -> Servidor

Mensaje: FILE_INFO_REQUEST

Se solicita información sobre el fichero cuyo nombre contiene una subcadena dada.

Servidor -> Cliente

Mensaje: FILE_NOT_FOUND

No se ha encontrado un fichero cuyo nombre tenga coincidencia con la subcadena recibida.

O bien durante la propia descarga:

Cliente -> Servidor:

Mensaje: GET_CHUNK

Se solicita un chunk que no existe, o un contenido de un fichero que ha sido modificado.

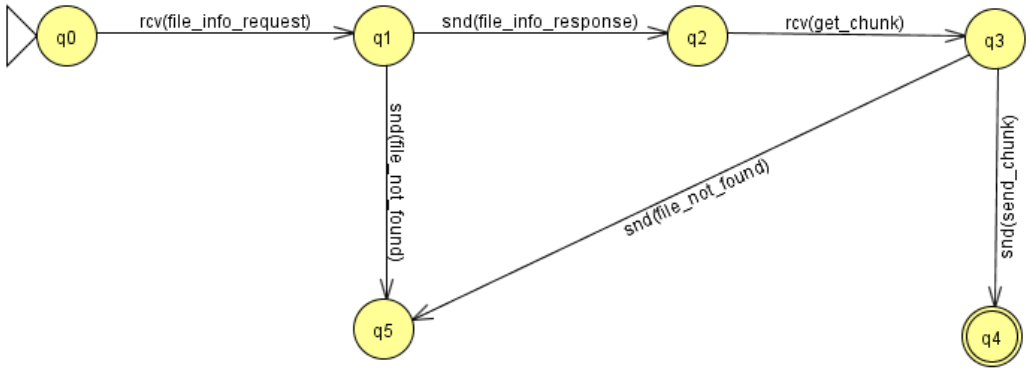
Servidor -> Cliente:

Mensaje: FILE_NOT_FOUND

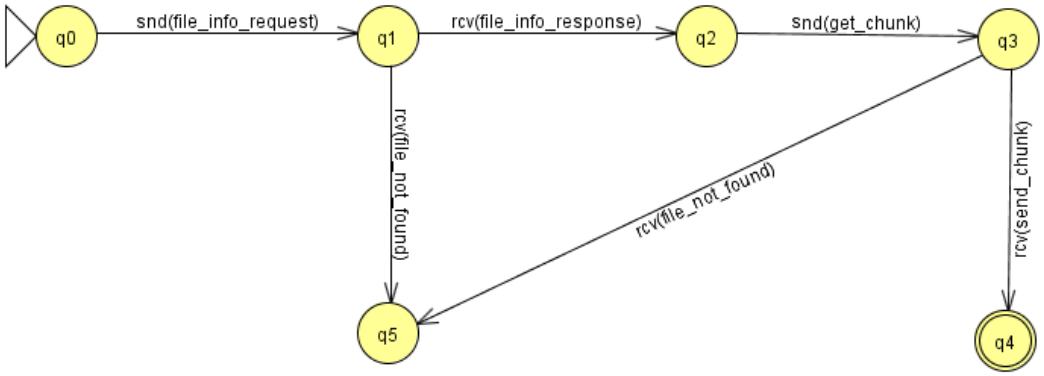
El servidor detecta el fallo, y responde con error.

Autómatas cliente y servidor

- Autómata Peer servidor de ficheros:



- Autómata Peer cliente de ficheros:



Mejoras implementadas

- Comando serve con puerto efímero:

  Se ha modificado el comportamiento del servidor de ficheros (NFServer) para que, en lugar de escuchar en un puerto fijo, por ejemplo, el 10.000, utilice un puerto efímero asignado automáticamente por el sistema operativo.
  Para ello en la clase NFServer se ha sustituido el valor de la constante PORT de 10000 a 0, del modo:

  public static final int PORT = 0;

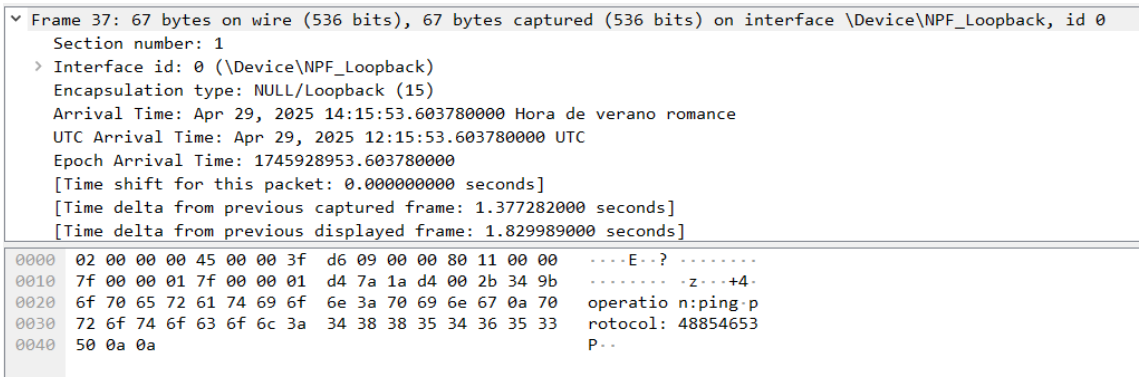
  En Java, cuando se crea un ServerSocket con puerto 0, el sistema asigna dinámicamente un puerto libre disponible.
  Para la consulta del puerto asignado tras la creación del servidor, se ha implementado el método getListeningPort(), que permite recuperar el número de puerto que está utilizando el servidor de ficheros.

Capturas de pantalla Wireshark

- Intercambio completo

```
 37 8.193612   127.0.0.1      127.0.0.1      UDP      67 54394 → 6868 Len=35
 38 8.214977   127.0.0.1      127.0.0.1      UDP      50 6868 → 54394 Len=18
 39 10.470487  127.0.0.1      127.0.0.1      UDP     383 54394 → 6868 Len=351
 40 10.494377  127.0.0.1      127.0.0.1      UDP      51 6868 → 54394 Len=19
145 13.982416  127.0.0.1      127.0.0.1      UDP      52 54394 → 6868 Len=20
146 13.994120  127.0.0.1      127.0.0.1      UDP     377 6868 → 54394 Len=345
201 23.920479  127.0.0.1      127.0.0.1      UDP      64 54394 → 6868 Len=32
202 23.922708  127.0.0.1      127.0.0.1      UDP      92 6868 → 54394 Len=60
```

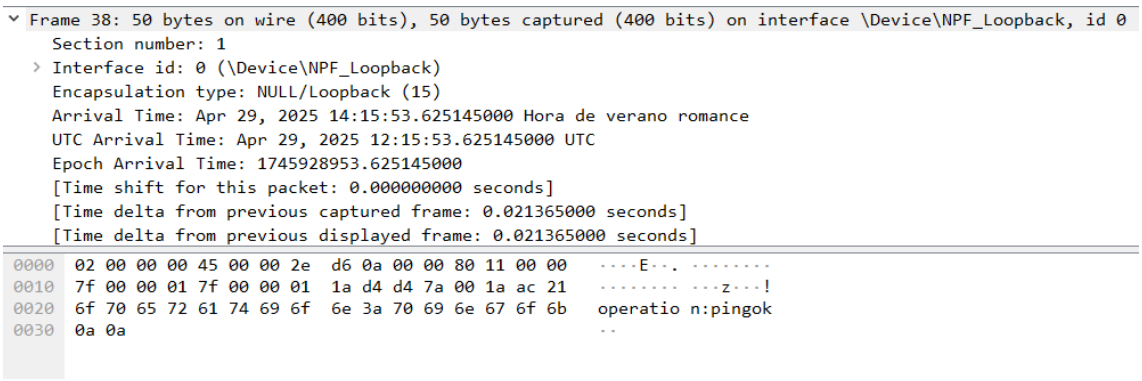
- Frame 37: ping



```
Frame 37: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface \Device\NPF_Loopback, id 0
    Section number: 1
  > Interface id: 0 (\Device\NPF_Loopback)
    Encapsulation type: NULL/Loopback (15)
    Arrival Time: Apr 29, 2025 14:15:53.603780000 Hora de verano romance
    UTC Arrival Time: Apr 29, 2025 12:15:53.603780000 UTC
    Epoch Arrival Time: 1745928953.603780000
    [Time shift for this packet: 0.000000000 seconds]
    [Time delta from previous captured frame: 1.377282000 seconds]
    [Time delta from previous displayed frame: 1.829989000 seconds]

0000  02 00 00 00 45 00 00 3f  d6 09 00 00 80 11 00 00   ····E··? ········
0010  7f 00 00 01 7f 00 00 01  d4 7a 1a d4 00 2b 34 9b   ········ ·z···+4·
0020  6f 70 65 72 61 74 69 6f  6e 3a 70 69 6e 67 0a 70   operatio n:ping·p
0030  72 6f 74 6f 63 6f 6c 3a  34 38 38 35 34 36 35 33   rotocol: 48854653
0040  50 0a 0a                                          P··
```

- Frame 38: pingok



```
Frame 38: 50 bytes on wire (400 bits), 50 bytes captured (400 bits) on interface \Device\NPF_Loopback, id 0
    Section number: 1
  > Interface id: 0 (\Device\NPF_Loopback)
    Encapsulation type: NULL/Loopback (15)
    Arrival Time: Apr 29, 2025 14:15:53.625145000 Hora de verano romance
    UTC Arrival Time: Apr 29, 2025 12:15:53.625145000 UTC
    Epoch Arrival Time: 1745928953.625145000
    [Time shift for this packet: 0.000000000 seconds]
    [Time delta from previous captured frame: 0.021365000 seconds]
    [Time delta from previous displayed frame: 0.021365000 seconds]

0000  02 00 00 00 45 00 00 2e  d6 0a 00 00 80 11 00 00   ····E···. ········
0010  7f 00 00 01 7f 00 00 01  1a d4 d4 7a 00 1a ac 21   ········ ···z···!
0020  6f 70 65 72 61 74 69 6f  6e 3a 70 69 6e 67 6f 6b   operatio n:pingok
0030  0a 0a                                             ··
```

- Frame 39: serve

```
Frame 39: 383 bytes on wire (3064 bits), 383 bytes captured (3064 bits) on interface \Device\NPF_Loopback, id 0
    Section number: 1
  > Interface id: 0 (\Device\NPF_Loopback)
    Encapsulation type: NULL/Loopback (15)
    Arrival Time: Apr 29, 2025 14:15:55.880655000 Hora de verano romance
    UTC Arrival Time: Apr 29, 2025 12:15:55.880655000 UTC
    Epoch Arrival Time: 1745928955.880655000
    [Time shift for this packet: 0.000000000 seconds]
    [Time delta from previous captured frame: 2.255510000 seconds]
```

```
0000  02 00 00 00 45 00 01 7b  d6 0b 00 00 80 11 00 00   ····E··{ ········
0010  7f 00 00 01 7f 00 00 01  d4 7a 1a d4 01 67 d9 f0   ········ ·z···g··
0020  6f 70 65 72 61 74 69 6f  6e 3a 73 65 72 76 65 0a   operatio n:serve·
0030  70 6f 72 74 3a 35 31 34  39 33 0a 66 69 6c 65 63   port:514 93·filec
0040  6f 75 6e 74 3a 34 0a 66  69 6c 65 6e 61 6d 65 3a   ount:4·f ilename:
0050  65 73 74 6f 66 61 64 6f  54 65 72 6e 65 72 61 2e   estofado Ternera.
0060  74 78 74 0a 73 69 7a 65  3a 31 31 32 38 0a 68 61   txt·size :1128·ha
0070  73 68 3a 31 61 63 63 37  63 62 31 66 64 61 63 63   sh:1acc7 cb1fdacc
0080  66 30 61 64 37 30 30 66  34 39 37 35 33 66 36 66   f0ad700f 49753f6f
0090  64 31 36 34 38 32 31 32  34 30 64 0a 66 69 6c 65   d1648212 40d·file
00a0  6e 61 6d 65 3a 70 72 75  65 62 61 2e 74 78 74 0a   name:pru eba.txt·
00b0  73 69 7a 65 3a 31 30 0a  68 61 73 68 3a 65 64 31   size:10· hash:ed1
```

- Frame 40: serveok

```
Frame 40: 51 bytes on wire (408 bits), 51 bytes captured (408 bits) on interface \Device\NPF_Loopback, id 0
    Section number: 1
  > Interface id: 0 (\Device\NPF_Loopback)
    Encapsulation type: NULL/Loopback (15)
    Arrival Time: Apr 29, 2025 14:15:55.904545000 Hora de verano romance
    UTC Arrival Time: Apr 29, 2025 12:15:55.904545000 UTC
    Epoch Arrival Time: 1745928955.904545000
    [Time shift for this packet: 0.000000000 seconds]
    [Time delta from previous captured frame: 0.023890000 seconds]
    [Time delta from previous displayed frame: 0.023890000 seconds]
```

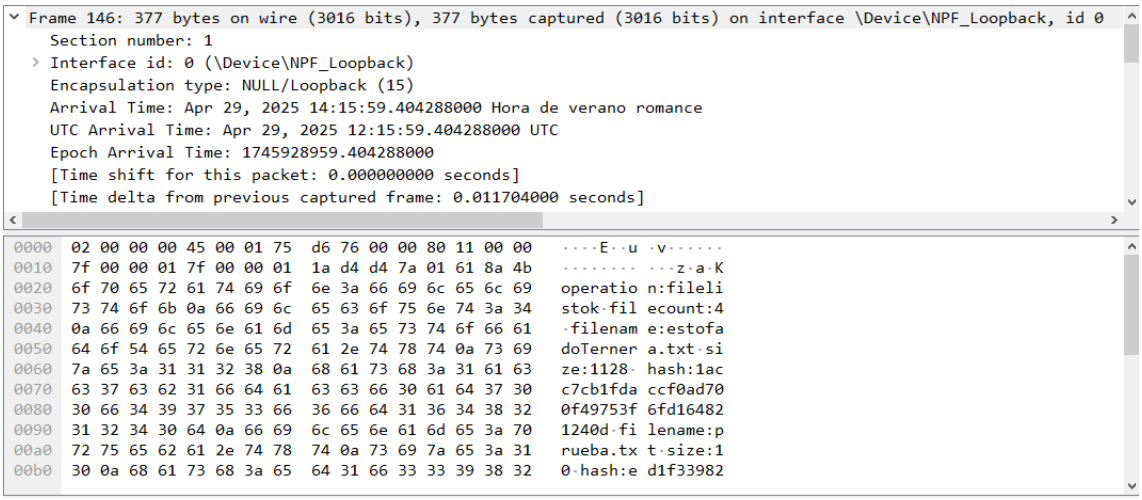
```
0000  02 00 00 00 45 00 00 2f  d6 0c 00 00 80 11 00 00   ····E··/ ········
0010  7f 00 00 01 7f 00 00 01  1a d4 d4 7a 00 1b 44 10   ········ ···z··D·
0020  6f 70 65 72 61 74 69 6f  6e 3a 73 65 72 76 65 6f   operatio n:serveo
0030  6b 0a 0a                                          k··
```

- Frame 145: filelist

```
Frame 145: 52 bytes on wire (416 bits), 52 bytes captured (416 bits) on interface \Device\NPF_Loopback, id 0
    Section number: 1
  > Interface id: 0 (\Device\NPF_Loopback)
    Encapsulation type: NULL/Loopback (15)
    Arrival Time: Apr 29, 2025 14:15:59.392584000 Hora de verano romance
    UTC Arrival Time: Apr 29, 2025 12:15:59.392584000 UTC
    Epoch Arrival Time: 1745928959.392584000
    [Time shift for this packet: 0.000000000 seconds]
    [Time delta from previous captured frame: 0.836263000 seconds]
    [Time delta from previous displayed frame: 3.488039000 seconds]
```

```
0000  02 00 00 00 45 00 00 30  d6 75 00 00 80 11 00 00   ····E··0 ·u······
0010  7f 00 00 01 7f 00 00 01  d4 7a 1a d4 00 1c 47 ad   ········ ·z····G·
0020  6f 70 65 72 61 74 69 6f  6e 3a 66 69 6c 65 6c 69   operatio n:fileli
0030  73 74 0a 0a                                       st··
```

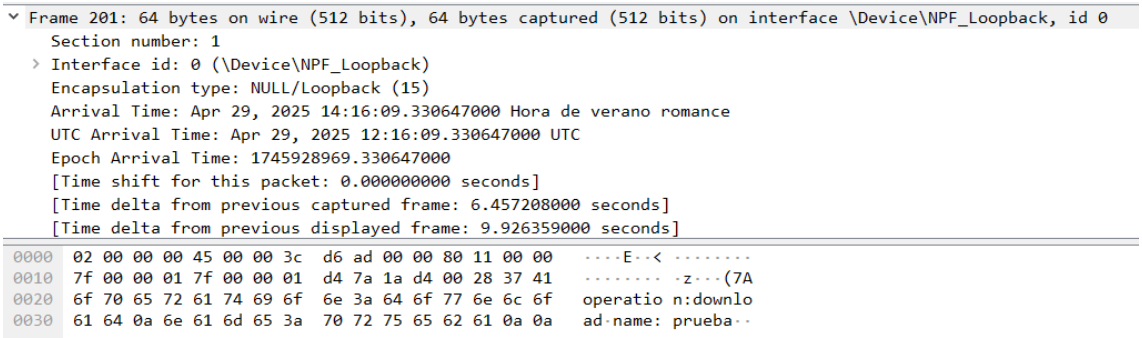
- Frame 146: filelistok



```
Frame 146: 377 bytes on wire (3016 bits), 377 bytes captured (3016 bits) on interface \Device\NPF_Loopback, id 0
   Section number: 1
 > Interface id: 0 (\Device\NPF_Loopback)
   Encapsulation type: NULL/Loopback (15)
   Arrival Time: Apr 29, 2025 14:15:59.404288000 Hora de verano romance
   UTC Arrival Time: Apr 29, 2025 12:15:59.404288000 UTC
   Epoch Arrival Time: 1745928959.404288000
   [Time shift for this packet: 0.000000000 seconds]
   [Time delta from previous captured frame: 0.011704000 seconds]
```

```
0000  02 00 00 00 45 00 01 75  d6 76 00 00 80 11 00 00   ····E··u ·v······
0010  7f 00 00 01 7f 00 00 01  1a d4 d4 7a 01 61 8a 4b   ········ ···z·a·K
0020  6f 70 65 72 61 74 69 6f  6e 3a 66 69 6c 65 6c 69   operatio n:fileli
0030  73 74 6f 6b 0a 66 69 6c  65 63 6f 75 6e 74 3a 34   stok·fil ecount:4
0040  0a 66 69 6c 65 6e 61 6d  65 3a 65 73 74 6f 66 61   ·filenam e:estofa
0050  64 6f 54 65 72 6e 65 72  61 2e 74 78 74 0a 73 69   doTerner a.txt·si
0060  7a 65 3a 31 31 32 38 0a  68 61 73 68 3a 31 61 63   ze:1128· hash:1ac
0070  63 37 63 62 31 66 64 61  63 63 66 30 61 64 37 30   c7cb1fda ccf0ad70
0080  30 66 34 39 37 35 33 66  36 66 64 31 36 34 38 32   0f49753f 6fd16482
0090  31 32 34 30 64 0a 66 69  6c 65 6e 61 6d 65 3a 70   1240d·fi lename:p
00a0  72 75 65 62 61 2e 74 78  74 0a 73 69 7a 65 3a 31   rueba.tx t·size:1
00b0  30 0a 68 61 73 68 3a 65  64 31 66 33 33 39 38 32   0·hash:e d1f33982
```

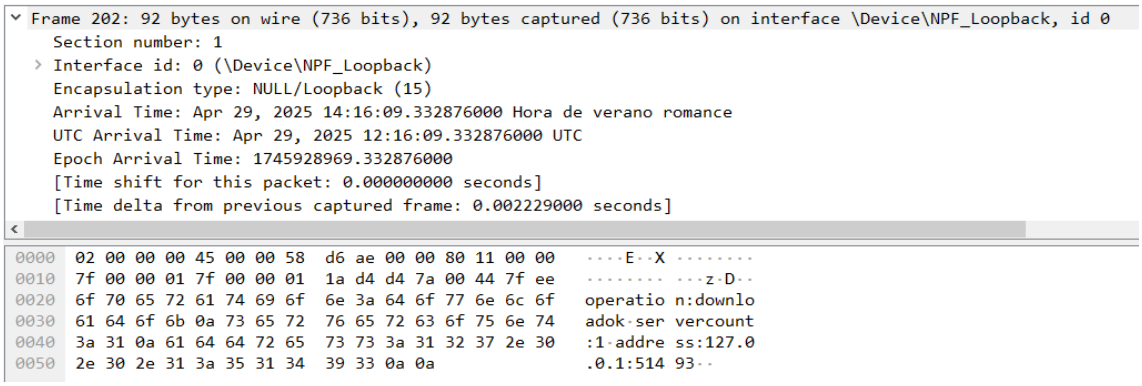
- Frame 201: download



```
Frame 201: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface \Device\NPF_Loopback, id 0
   Section number: 1
 > Interface id: 0 (\Device\NPF_Loopback)
   Encapsulation type: NULL/Loopback (15)
   Arrival Time: Apr 29, 2025 14:16:09.330647000 Hora de verano romance
   UTC Arrival Time: Apr 29, 2025 12:16:09.330647000 UTC
   Epoch Arrival Time: 1745928969.330647000
   [Time shift for this packet: 0.000000000 seconds]
   [Time delta from previous captured frame: 6.457208000 seconds]
   [Time delta from previous displayed frame: 9.926359000 seconds]
```

```
0000  02 00 00 00 45 00 00 3c  d6 ad 00 00 80 11 00 00   ····E··< ········
0010  7f 00 00 01 7f 00 00 01  d4 7a 1a d4 00 28 37 41   ········ ·z···(7A
0020  6f 70 65 72 61 74 69 6f  6e 3a 64 6f 77 6e 6c 6f   operatio n:downlo
0030  61 64 0a 6e 61 6d 65 3a  70 72 75 65 62 61 0a 0a   ad·name: prueba··
```

- Frame 202: downloadok



```
Frame 202: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface \Device\NPF_Loopback, id 0
   Section number: 1
 > Interface id: 0 (\Device\NPF_Loopback)
   Encapsulation type: NULL/Loopback (15)
   Arrival Time: Apr 29, 2025 14:16:09.332876000 Hora de verano romance
   UTC Arrival Time: Apr 29, 2025 12:16:09.332876000 UTC
   Epoch Arrival Time: 1745928969.332876000
   [Time shift for this packet: 0.000000000 seconds]
   [Time delta from previous captured frame: 0.002229000 seconds]
```

```
0000  02 00 00 00 45 00 00 58  d6 ae 00 00 80 11 00 00   ····E··X ········
0010  7f 00 00 01 7f 00 00 01  1a d4 d4 7a 00 44 7f ee   ········ ···z·D··
0020  6f 70 65 72 61 74 69 6f  6e 3a 64 6f 77 6e 6c 6f   operatio n:downlo
0030  61 64 6f 6b 0a 73 65 72  76 65 72 63 6f 75 6e 74   adok·ser vercount
0040  3a 31 0a 61 64 64 72 65  73 73 3a 31 32 37 2e 30   :1·addre ss:127.0
0050  2e 30 2e 31 3a 35 31 34  39 33 0a 0a               .0.1:514 93··
```

Conclusiones

Este proyecto ha sido, sin duda, uno de los mayores retos a los que nos hemos enfrentado. Al principio, la práctica parecía inmanejable: múltiples capas de comunicación, diferentes protocolos, servidores concurrentes, sockets, hilos… era difícil saber por donde empezar. La sensación inicial era de estar completamente perdido, sin una visión clara de cómo encajar entre sí todas las piezas del sistema.

Tras muchas horas frente al ordenador, todo comenzó poco a poco a tomar forma. El sistema fue cobrando sentido a medida que se implementaban las distintas partes y se veían los primeros resultados. Uno de los mayores desafíos fue la implementación de la comunicación TCP entre peers, en particular el proceso de descarga de ficheros. Diseñar un protocolo robusto, dividir los archivos en chunks, gestionar múltiples conexiones y asegurar la integridad de los datos requirió un gran esfuerzo, además de muchas iteraciones y pruebas. La depuración de errores en tiempo de ejecución y la revisión constante de que todo funcionara correctamente, también supuso un reto constante durante el desarrollo.

En resumen, ha sido una práctica tan compleja como gratificante. Más allá del código, nos ha enseñado a ser más pacientes, a planificar mejor, a depurar con cabeza y, sobre todo, a confiar en que, aunque algo al principio parezca imposible, con trabajo y persistencia puede salir adelante.