Diseño de proyecto

NanoFiles

José Javier MéndezLagunas

Alejandro Segura García

Curso 2023/2024

ÍNDICE

1. Introducción

En el documento se especifica el diseño de dos protocolos de comunicación el primero Cliente - Directorio que operará sobre un protocolo de nivel de transporte no confiable (UDP) y el segundo entre cliente y servidor de ficheros, con un protocolo de nivel de transporte confiable (TCP).

---

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

A la hora de definir el protocolo de comunicación con el Directorio, vamos a utilizar mensajes textuales. Los mensajes de confirmación en caso de fallo, serían iguales modificando el status a Fail.

Tipos de mensaje

Mensajes solo código de operación

Operación
op

Mensajes con datos

Operación	**Nickname**
op	**nick**

Mensaje: **login**
Sentido de la comunicación: Cliente→ Directorio
Descripción: Este mensaje lo envía el cliente al Directorio para solicitar iniciar sesión y registrar su nickname.
Ejemplo:

op: login
nick: JJ
/n

Mensaje: **confirmación login**
Sentido de la comunicación: Directorio→ Cliente
Descripción: Este mensaje confirma si el inicio de sesión se realizó, de ser así proporciona la Key.
Ejemplo acierto:

Estado: Success
Salida: Login successful
Key: 0000001
\n

Mensaje: **logout**
Sentido de la comunicación: Cliente→ Directorio
Descripción: Este mensaje lo envía el cliente de NanoFiles al Directorio para solicitar cerrar la sesión actual y borrar tanto el nickname como la Key asociadas.
Ejemplo:

op: logout
nick: Key
\n

Mensaje: **confirmación logout**
Sentido de la comunicación: Directorio→ Cliente
Este mensaje confirma si el logout se realizó con éxito.
Ejemplo acierto:

Estado: Success
Salida: Logout Successful
\n

Mensaje: **userlist**
Sentido de la comunicación: Cliente→ Directorio
Descripción: Este mensaje lo envía el cliente de NanoFiles al Directorio para solicitar la lista de usuarios registrados.
Ejemplo:

op: userlist
\n

Mensaje: **confirmación userlist**
Sentido de la comunicación: Directorio→ Cliente
Este mensaje confirma si el userlist se realizó con éxito
Ejemplo acierto:

Estado: Success
Salida: (username1,username2,username3…)
\n

Mensaje: **fgserve**
Sentido de la comunicación: Cliente→ Directorio
Descripción: Este mensaje lo envía el cliente de NanoFiles al Directorio para solicitar darse de alta como servidor.
Ejemplo:

op: fgserve
\n

Mensaje: **bgserve**
Sentido de la comunicación: Cliente→ Directorio
Descripción: Este mensaje lo envía el cliente de NanoFiles al Directorio para solicitar la baja como servidor de ficheros.
Ejemplo:

op: bgserver
\n

Mensaje: **confirmación bgserve**
Sentido de la comunicación: Directorio→ Cliente
Este mensaje confirma si el bgserve ha sido un éxito
Ejemplo acierto:

Estado: Success
Salida: bgserve Successful
\n

Mensaje: **stopserver**
Sentido de la comunicación: Cliente→ Directorio
Descripción: Este mensaje lo envía el cliente de NanoFiles al Directorio para solicitar la baja como servidor de ficheros.
Ejemplo:

op: stopserver

Mensaje: **confirmación stopserver**
Sentido de la comunicación: Directorio→ Cliente
Este mensaje confirma si el stopserver ha sido un éxito
Ejemplo acierto:

Estado: Success
Salida: Stopserver Successful
\n

3. Formato de los mensajes del protocolo de comunicación entre cliente y servidor de ficheros

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

Sentido de la comunicación: Servidor de ficheros → Cliente

Descripción: Este mensaje lo envía el par servidor de ficheros al par cliente de fichero para indicar que el mensaje enviado previamente estaba identificado por un código no válido.

Ejemplo:

OpCode
0

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

Sentido de la comunicación: Cliente → Servidor de fichero

Descripción: Este mensaje lo envía el par cliente al par servidor de ficheros para descargar un fichero identificado por su hash.

Ejemplo:

OpCode	**Hash**	**Length**
1	Hash	Length

Mensaje: **download_OK** (opcode = 2)

Sentido de la comunicación: Servidor de Cliente → fichero

Descripción: Este mensaje lo envía el cliente al par servidor de ficheros para descargar un fichero identificado por su código hash.

Ejemplo:

OpCode	**Hash**	**Length**	**Data**
2	Hash	Length	Data

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

Sentido de la comunicación: Cliente → Servidor de fichero

Descripción: Este mensaje lo envía el par cliente al par servidor de ficheros para preguntar el tamaño de un fichero

Ejemplo:

OpCode	**Hash**
3	Hash

Mensaje: tam (opcode = 4)

Sentido de la comunicación: Servidor de Cliente → fichero

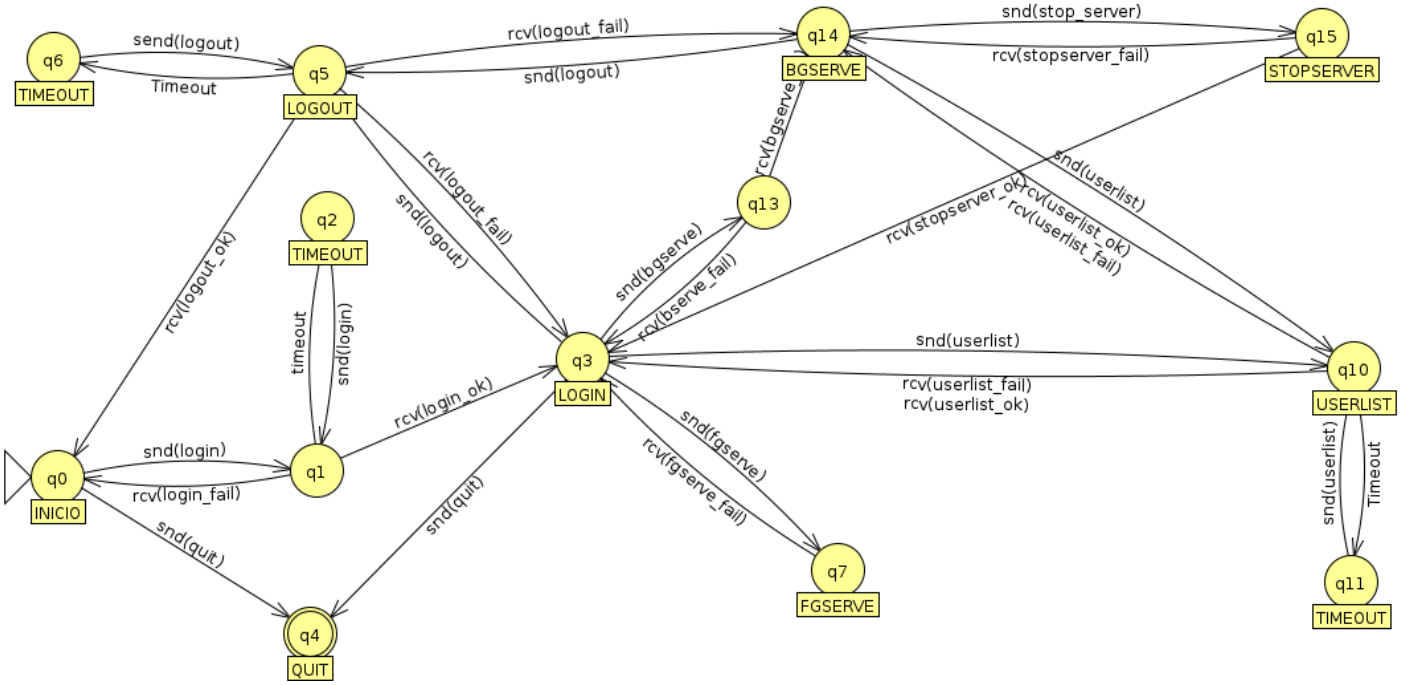
Descripción: Este mensaje lo envía el cliente al par servidor de ficheros para devolver el tamaño de un fichero.

Ejemplo:

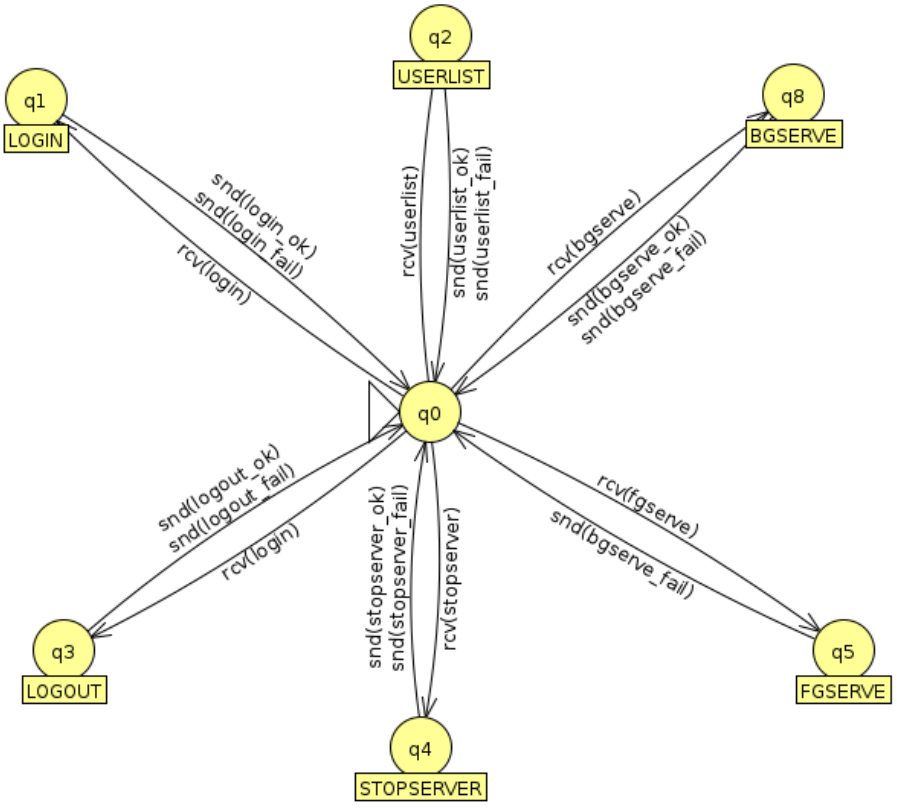
OpCode	**Length**
4	Length

4. Autómatas de protocolo

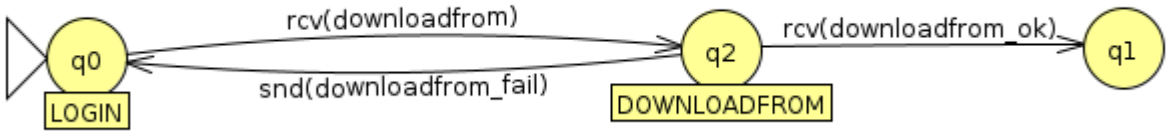
- Autómata Cliente de directorio



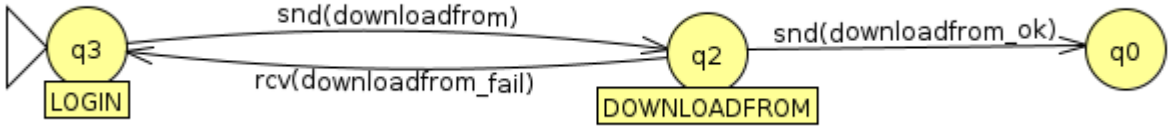
- Autómata Servidor de ficheros



- **Autómata Servidor de directorio**



- **Autómata Cliente de ficheros**



5. Mejoras

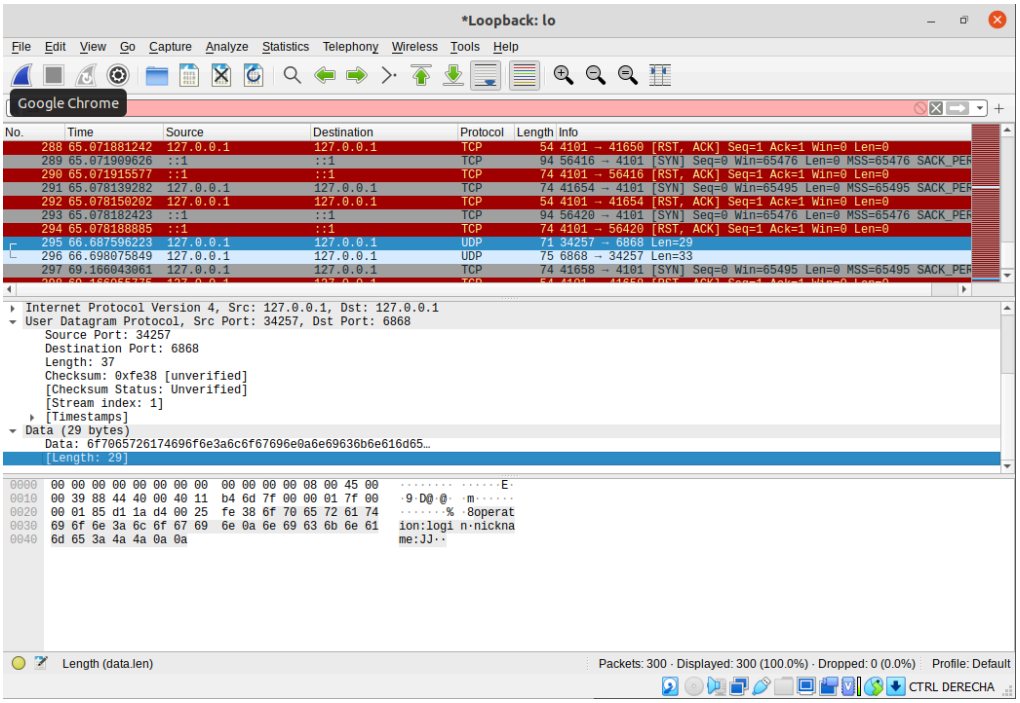
- **fgserve puerto variable 0,5 punto(s)**
- **bgserve secuencial 1 punto(s)**
- **stopserver 0,5 punto(s)**
- **bgserve puerto efímero 0,5 punto(s)**

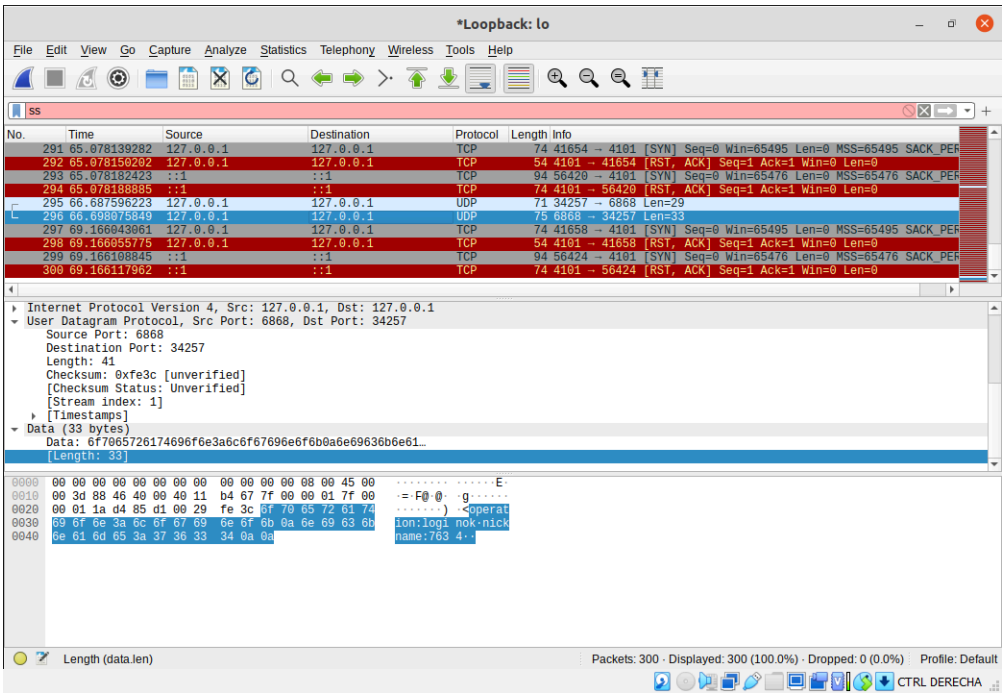
Para la mejora de fgserve de puerto variable hemos que cuando se intente usar un puerto que no esté disponible que le sume 1 a dicho puerto hasta que si se pueda bindear, entonces saldrá del bucle y se asigara este, respecto al bgserve secuencial hemos hecho uso de la función .getLocalPort, que asigna de forma automática al bgserve cualquier puerto disponible. Para la mejora de stopserver hemos añadido la función stopServer() a la clase NFServer para disponer de esta.

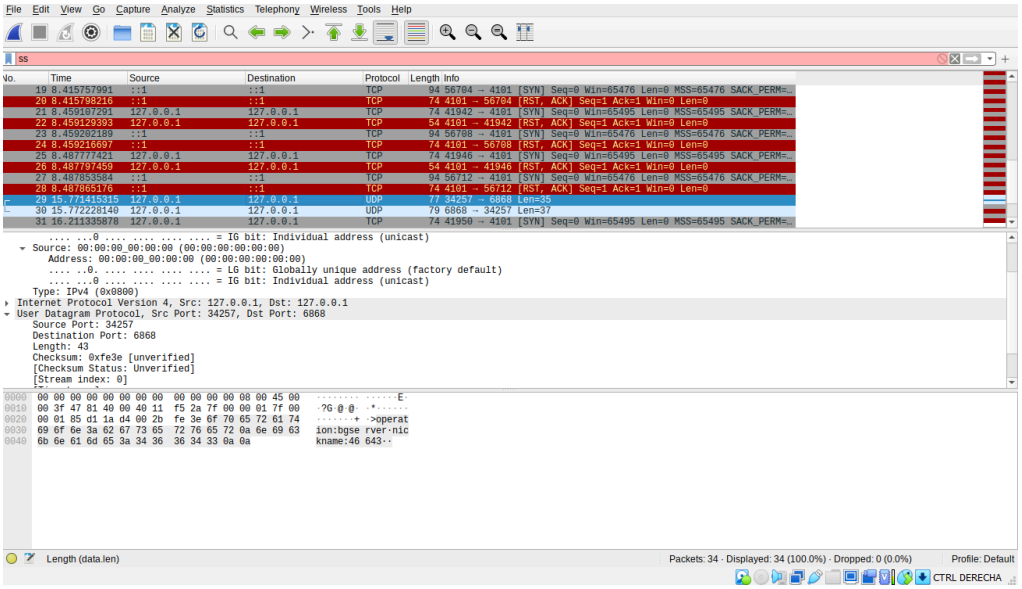
Y por último para la mejora de bgserve puerto efímero se crea el socket con puerto a 0. Esto ya hace que el propio socket elija un número aleatorio. Después simplemente comprobamos que es un puerto válido.

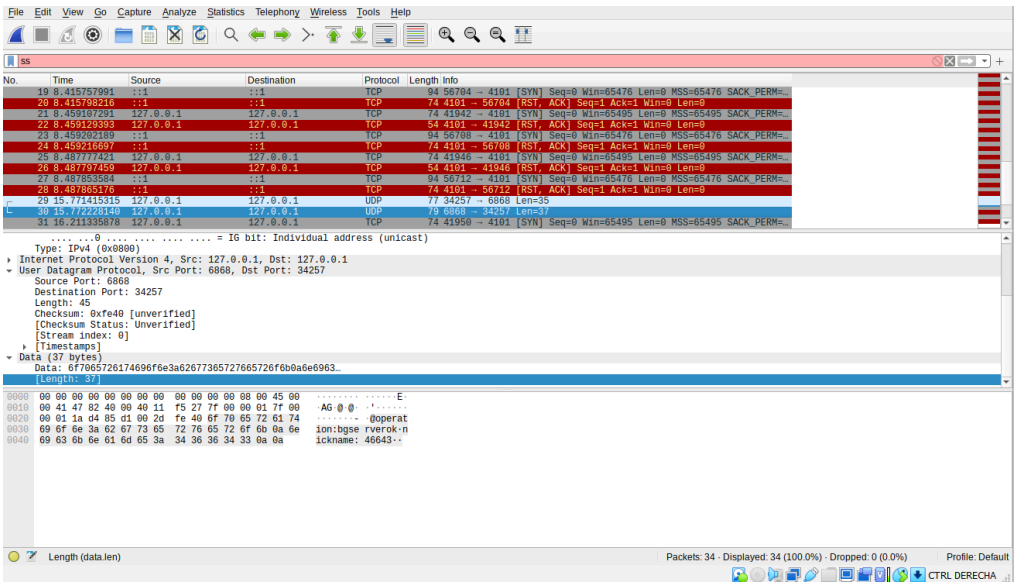
6. Capturas WireShark

A continuación se encuentra la siguiente secuencia de mensajes capturada desde WireShark, inicialmente el cliente envía login al directorio, este le contesta con loginok para despues enviarle bgserve al directorio y que este le devuelva bgserveok.






File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ss
No. Time Source Destination Protocol Length Info
19 8.415757991 ::1 ::1 TCP 94 56704 → 4101 [SYN] Seq=0 Win=65476 Len=0 MSS=65476 SACK_PERM=…
20 8.415798216 ::1 ::1 TCP 74 4101 → 56704 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
21 8.459107291 127.0.0.1 127.0.0.1 TCP 74 41942 → 4101 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=…
22 8.459129393 127.0.0.1 127.0.0.1 TCP 54 4101 → 41942 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
23 8.459202189 ::1 ::1 TCP 94 56708 → 4101 [SYN] Seq=0 Win=65476 Len=0 MSS=65476 SACK_PERM=…
24 8.459216697 ::1 ::1 TCP 74 4101 → 56708 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
25 8.487777421 127.0.0.1 127.0.0.1 TCP 74 41946 → 4101 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=…
26 8.487797459 127.0.0.1 127.0.0.1 TCP 54 4101 → 41946 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
27 8.487853584 ::1 ::1 TCP 94 56712 → 4101 [SYN] Seq=0 Win=65476 Len=0 MSS=65476 SACK_PERM=…
28 8.487865176 ::1 ::1 TCP 74 4101 → 56712 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
29 15.771415315 127.0.0.1 127.0.0.1 UDP 77 34257 → 6868 Len=35
30 15.772228140 127.0.0.1 127.0.0.1 UDP 79 6868 → 34257 Len=37
31 16.211335878 127.0.0.1 127.0.0.1 TCP 74 41950 → 4101 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=…
....0 .... = IG bit: Individual address (unicast)
Source: 00:00:00_00:00:00 (00:00:00:00:00:00)
Address: 00:00:00_00:00:00 (00:00:00:00:00:00)
.... ..0. .... = LG bit: Globally unique address (factory default)
....0 .... = IG bit: Individual address (unicast)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
User Datagram Protocol, Src Port: 34257, Dst Port: 6868
Source Port: 34257
Destination Port: 6868
Length: 43
Checksum: 0xfe3e [unverified]
[Checksum Status: Unverified]
[Stream index: 0]
0000 00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00 ..... E.
0010 00 3f 47 81 40 00 40 11 f5 2a 7f 00 00 01 7f 00 .?G.@.@. .*.....
0020 00 01 85 d1 1a d4 00 2b fe 3e 6f 70 65 72 61 74 ..... + .>operat
0030 69 6f 6e 3a 62 67 73 65 72 76 65 72 0a 6e 69 63 ion:bgse rver.nic
0040 6b 6e 61 6d 65 3a 34 36 36 34 33 0a 0a kname:46 643..
Length (data.len)
Packets: 34 · Displayed: 34 (100.0%) · Dropped: 0 (0.0%)
Profile: Default
CTRL DERECHA


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ss
No. Time Source Destination Protocol Length Info
19 8.415757991 ::1 ::1 TCP 94 56704 → 4101 [SYN] Seq=0 Win=65476 Len=0 MSS=65476 SACK_PERM=…
20 8.415798216 ::1 ::1 TCP 74 4101 → 56704 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
21 8.459107291 127.0.0.1 127.0.0.1 TCP 74 41942 → 4101 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=…
22 8.459129393 127.0.0.1 127.0.0.1 TCP 54 4101 → 41942 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
23 8.459202189 ::1 ::1 TCP 94 56708 → 4101 [SYN] Seq=0 Win=65476 Len=0 MSS=65476 SACK_PERM=…
24 8.459216697 ::1 ::1 TCP 74 4101 → 56708 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
25 8.487777421 127.0.0.1 127.0.0.1 TCP 74 41946 → 4101 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=…
26 8.487797459 127.0.0.1 127.0.0.1 TCP 54 4101 → 41946 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
27 8.487853584 ::1 ::1 TCP 94 56712 → 4101 [SYN] Seq=0 Win=65476 Len=0 MSS=65476 SACK_PERM=…
28 8.487865176 ::1 ::1 TCP 74 4101 → 56712 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
29 15.771415315 127.0.0.1 127.0.0.1 UDP 77 34257 → 6868 Len=35
30 15.772228140 127.0.0.1 127.0.0.1 UDP 79 6868 → 34257 Len=37
31 16.211335878 127.0.0.1 127.0.0.1 TCP 74 41950 → 4101 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=…
....0 .... = IG bit: Individual address (unicast)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
User Datagram Protocol, Src Port: 6868, Dst Port: 34257
Source Port: 6868
Destination Port: 34257
Length: 45
Checksum: 0xfe40 [unverified]
[Checksum Status: Unverified]
[Stream index: 0]
[Timestamps]
Data (37 bytes)
Data: 6f706572617469 6f6e3a626773657265 72 6f6b0a6e6963…
[Length: 37]
0000 00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00 ..... E.
0010 00 41 47 82 40 00 40 11 f5 27 7f 00 00 01 7f 00 .AG.@.@. .'.....
0020 00 01 1a d4 85 d1 00 2d fe 40 6f 70 65 72 61 74 ..... - .@operat
0030 69 6f 6e 3a 62 67 73 65 72 76 65 72 6f 6b 0a 6e ion:bgse rverok.n
0040 69 63 6b 6e 61 6d 65 3a 34 36 36 34 33 0a 0a ickname: 46643..
Length (data.len)
Packets: 34 · Displayed: 34 (100.0%) · Dropped: 0 (0.0%)
Profile: Default
CTRL DERECHA

7. Conclusiones

El proyecto es una manera práctica y directa de poder aplicar los conocimientos vistos previamente en teoría y además darte una perspectiva más real que únicamente ejercicios teóricos sobre lo relacionado con la asignatura. Además, es una buena actividad para aprender sobre diseño de protocolos y la importancia de la eficiencia dada por un buen diseño y la división por capas.