

Seminario 2

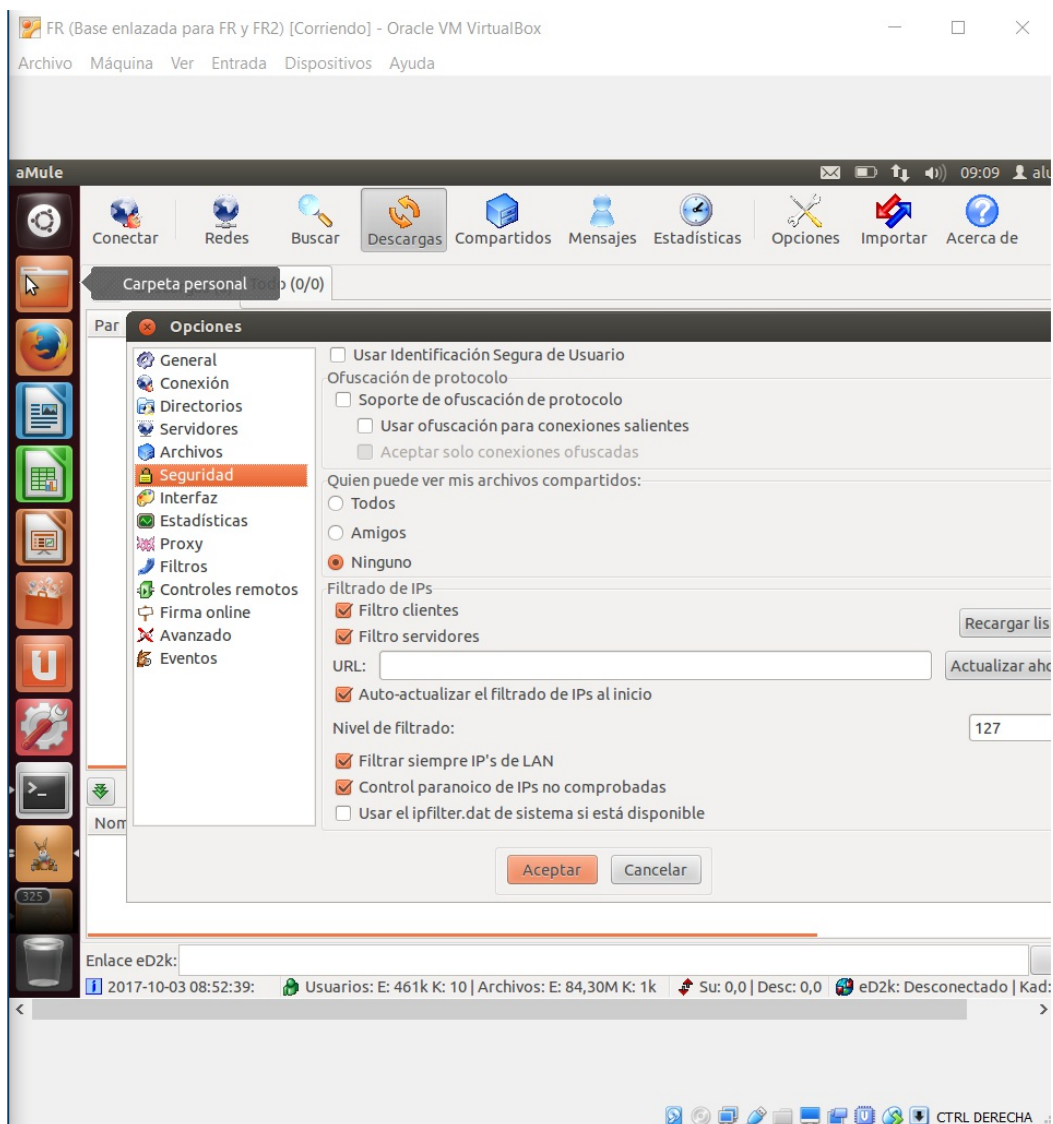
En este seminario se nos pide que realicemos una captura de tráfico de paquetes mientras realizamos una descarga de la imagen de ubuntu mediante el programa eMule.

1º Configuración de eMule.

Esta configuración la realizamos en la misma clase, de mano del profesor. Abrimos la pestaña "Opciones"-->"Seguridad" y desmarcamos las opciones:

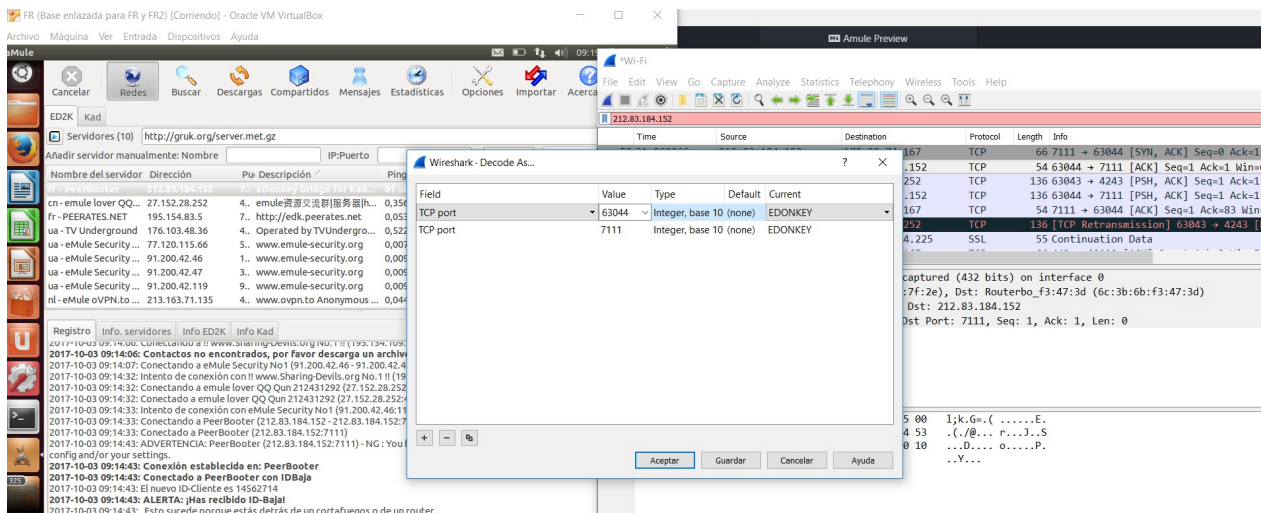
- Usar identificación Segura de Usuario.
- Usar ofuscación para conexiones salientes.
- Soporte de ofuscación de protocolo.

Esto lo hacemos, como bien explicamos en clase, para que no haya lugar a confusión a la hora de analizar los paquetes. Ya que si no podríamos llegar a confundirnos con esos paquetes "falsos" que envía para camuflar la conexión.



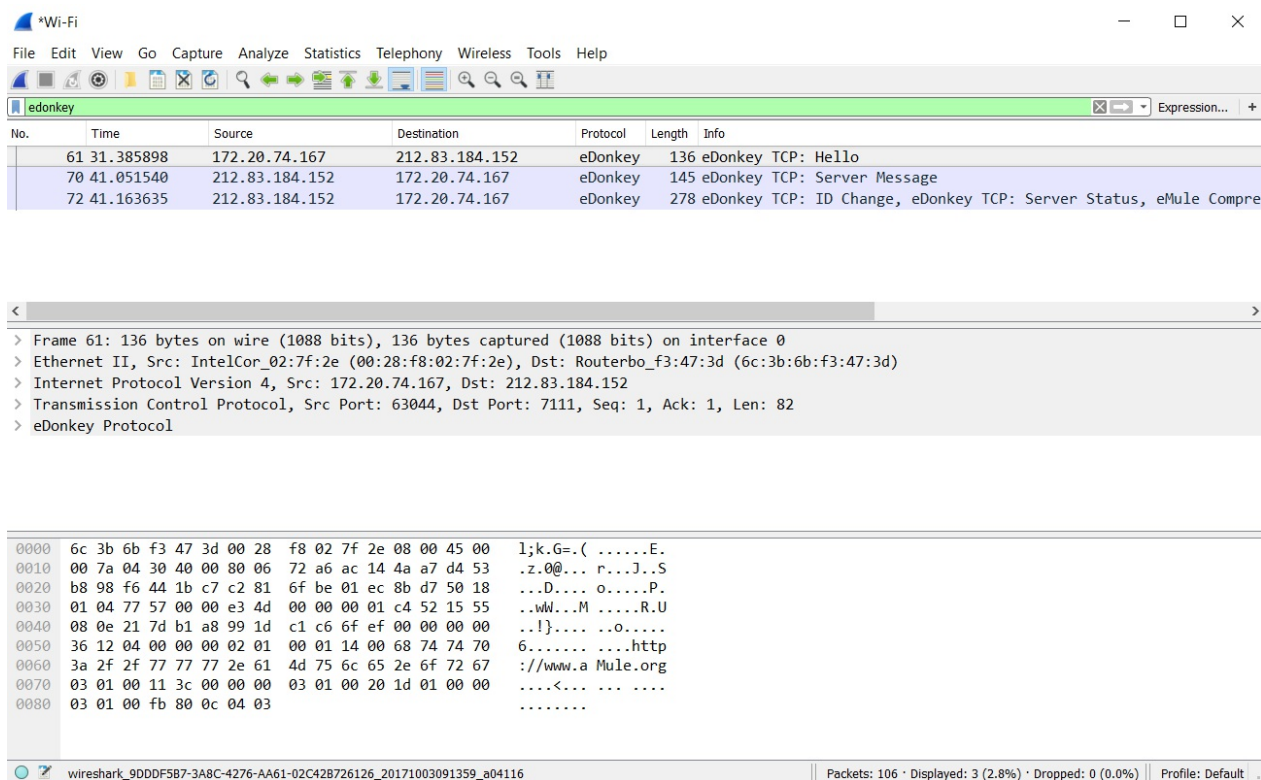
2º Inicializar el programa y comprobar conexiones

Acabamos de iniciar Wireshark, lo hemos puesto en marcha. Hemos puesto a buscar dirección de conexión, lo ha conseguido y este es el resultado.



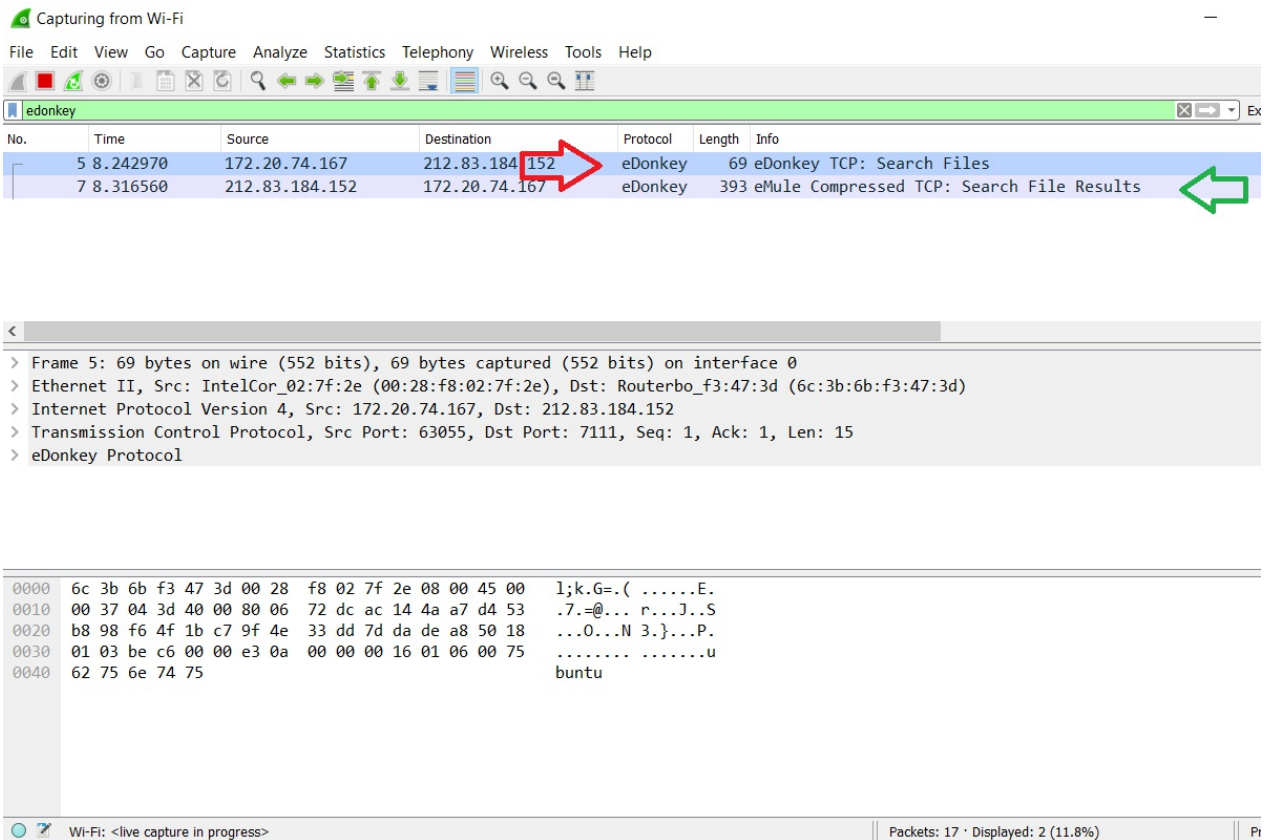
Hemos comprobado que hoy nos ha conectado a una IP(212.83.184.152) por y escucha en dos puertos el 63044 y el 7111, por defecto Wireshark es capaz de reconocer los paquetes eDonkey, pero estos están camuflados para evitar que las ISP's hagan de las suyas. Por tanto una vez analizados y sabiendo cuales son las IP's y los puertos de conexión, podemos decirle a nuestro Wireshark cuáles son los paquetes que debe reconocer como eDonkey.

Realizamos pues, la búsqueda de paquetes eDonkey y el resultado es:



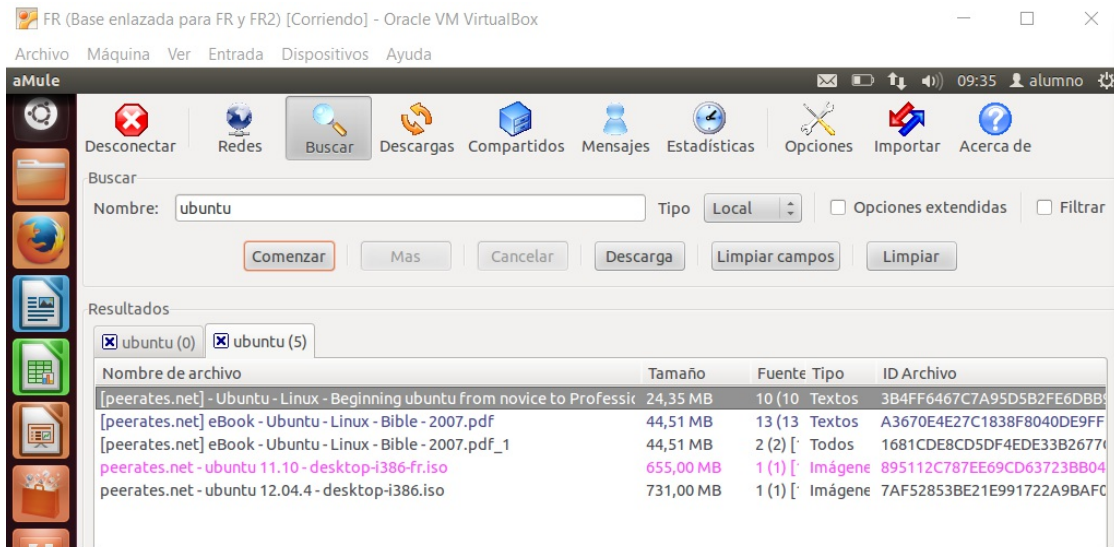
En esta foto podemos comprobar los mensajes de conexión:
 1º Hello: en el que nuestra máquina "saluda" a la dirección de destino.
 2º La respuesta: en la que el "servidor" nos responde dandonos una ID para nuestra conexión, y nos avisa de que abramos ciertos puertos o nuestra ID sera baja.

A continuación, hemos buscado una imagen ubuntu y estas son la petición de busqueda(ROJO) y la respuesta de: "archivo encontrado"(VERDE).



3º Conexión establecida y descarga

Vamos a descargar la imagen ubuntu y para ello hemos escogido una con varias fuentes para comprobar que ocurre.



Lo primero que comprobamos gracias a Wireshark es que hacemos dos ping de Hello a 2 direcciones distintas: 79.55.38.194 y 117.17.187.118. Aunque solo una de las dos nos responde: 117.17.187.118

Capturing from Wi-Fi

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

edonkey

No.	Time	Source	Destination	Protocol	Length	Info
5	8.242970	172.20.74.167	212.83.184.152	eDonkey	69	eDonkey TCP: Search Files
7	8.316560	212.83.184.152	172.20.74.167	eDonkey	393	eMule Compressed TCP: Search File Results
74	94.046792	172.20.74.167	212.83.184.152	eDonkey	80	eDonkey TCP: Get Sources
79	94.119863	212.83.184.152	172.20.74.167	eDonkey	107	eDonkey TCP: Found Sources
96	94.360634	172.20.74.167	79.55.38.194	eDonkey	167	eDonkey TCP: Hello
7148	111.745327	172.20.74.167	117.17.187.118	eDonkey	167	eDonkey TCP: Hello
8007	112.955541	117.17.187.118	172.20.74.167	eDonkey	170	eDonkey TCP: Hello Answer
8015	112.997241	172.20.74.167	117.17.187.118	eDonkey	96	eMule Extensions TCP: MultiPacketExt
8242	113.318211	117.17.187.118	172.20.74.167	eDonkey	77	eMule Extensions TCP: Hello
8593	113.765505	117.17.187.118	172.20.74.167	eDonkey	185	eMule Extensions TCP: MultiPacket Answer
8598	113.779572	172.20.74.167	117.17.187.118	eDonkey	77	eMule Extensions TCP: Hello
8846	114.100756	172.20.74.167	117.17.187.118	eDonkey	76	eDonkey TCP: Slot Request

> Frame 96: 167 bytes on wire (1336 bits), 167 bytes captured (1336 bits) on interface 0

> Ethernet II, Src: IntelCor_02:7f:2e (00:28:f8:02:7f:2e), Dst: Routerbo_f3:47:3d (6c:3b:6b:f3:47:3d)

> Internet Protocol Version 4, Src: 172.20.74.167, Dst: 79.55.38.194

> Transmission Control Protocol, Src Port: 63058, Dst Port: 4662, Seq: 1, Ack: 1, Len: 113

> eDonkey Protocol

```

0000 6c 3b 6b f3 47 3d 00 28 f8 02 7f 2e 08 00 45 00 l;k.G=( .....E.
0010 00 99 4a e4 40 00 80 06 42 c6 ac 14 4a a7 4f 37 ..J.G... B...J.07
0020 26 c2 f6 52 12 36 3a cf 37 ce 6e 95 90 7a 50 18 &..R.6:. 7.n..zP.
0030 01 04 01 cc 00 00 e3 6c 00 00 00 01 10 c4 52 15 .....l .....R.
0040 55 08 0e 21 7d b1 a8 99 1d c1 c6 6f ef 70 69 de U..!}... ..o.pi.
0050 00 36 12 07 00 00 00 02 01 00 01 14 00 68 74 74 .6..... ..htt
0060 70 3a 2f 2f 77 77 77 2e 61 4d 75 6c 65 2e 6f 72 p://www. aMule.or
0070 67 03 01 00 11 3c 00 00 00 03 01 00 f9 40 12 00 g.....<... ..@.
0080 00 03 01 00 fb 80 0c 04 03 03 01 00 fa 16 32 10 .....2

```

En esta foto se muestra como la dirección 117.17.187.118 nos está dando los paquetes de la imagen solicitada.

Capturing from Wi-Fi

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

edonkey

No.	Time	Source	Destination	Protocol	Length	Info
8593	113.765505	117.17.187.118	172.20.74.167	eDonkey	185	eMule Extensions TCP: MultiPacket Answer
8598	113.779572	172.20.74.167	117.17.187.118	eDonkey	77	eMule Extensions TCP: Hello
8846	114.100756	172.20.74.167	117.17.187.118	eDonkey	76	eDonkey TCP: Slot Request
9224	114.624598	117.17.187.118	172.20.74.167	eDonkey	60	eDonkey TCP: Slot Given
9289	114.719512	172.20.74.167	117.17.187.118	eDonkey	100	eDonkey TCP: Request Parts
9819	115.436729	117.17.187.118	172.20.74.167	eDonkey	1238	eMule Extensions TCP: Data Compressed
10129	115.815386	117.17.187.118	172.20.74.167	eDonkey	1417	eMule Extensions TCP: Data Compressed [TCP segment of a reassembled PDU]
21851	131.915970	172.20.74.167	212.83.184.152	eDonkey	183	eMule Compressed TCP: Offer Files
27140	499.965735	172.20.74.167	151.32.127.61	eDonkey	167	eDonkey TCP: Hello
27141	500.050595	151.32.127.61	172.20.74.167	eDonkey	143	eDonkey TCP: Hello Answer
27201	537.844330	172.20.74.167	2.225.188.162	eDonkey	85	KAdu UDP: Unknown
27308	578.007882	80.116.218.7	172.20.74.167	eDonkey	91	KAdu UDP: KADEMLIA ETREMAILFD_RES

8 Reassembled TCP Segments (10725 bytes): #9810(1363), #9811(1363), #9812(1363), #9813(1363), #9814(1363), #9817(1363), #9818(1363), #9819(1184)

[Frame: 9810, payload: 0-1362 (1363 bytes)]

[Frame: 9811, payload: 1363-2725 (1363 bytes)]

[Frame: 9812, payload: 2726-4088 (1363 bytes)]

[Frame: 9813, payload: 4089-5451 (1363 bytes)]

[Frame: 9814, payload: 5452-6814 (1363 bytes)]

[Frame: 9817, payload: 6815-8177 (1363 bytes)]

[Frame: 9818, payload: 8178-9500 (1363 bytes)]

[Frame: 9819, payload: 9541-10724 (1184 bytes)]

[Segment count: 8]

[Reassembled TCP length: 10725]

[Reassembled TCP Data: c5e0290000403b4ff6467c7a95d5b2fe6db9548023f0020...]

eDonkey Protocol

```

0000 c5 e0 29 00 00 40 3b 4f f6 46 7c 7a 95 d5 b2 fe ..).@;0 .f|z...
0010 6d bb 95 48 02 3f 00 20 1c 00 42 c6 02 00 78 da m..H.P. .B...X
0020 14 9a 65 58 94 5d 17 85 45 69 09 89 21 15 14 2d ..eX].Ei...S
0030 0c a1 bb 11 10 10 a4 e3 61 e8 ce 6e 41 42 49 e9 ..c...a..nB1
0040 72 40 5a 3a 66 80 19 ba 53 ba a4 86 6e 00 92 ee r@Z:f...S...n...
0050 fe de ef ef fe 79 fe 5a d7 ba d7 be ce b3 ee dd .....y.Z .....
0060 8c 04 94 1e 09 3c d8 14 54 e6 7b 7f 6a 14 e7 2d .....<.T.(.}.
0070 81 85 17 c6 40 a5 26 f3 c7 98 d7 a7 de cc 21 d5 A..@.&.....I.
0080 65 5c 4b 07 b7 f4 5a 0e 61 1b 09 ae 60 f3 90 04 .(Ko;Z.a.....
0090 84 0f cc 50 33 31 3d 35 0e 51 51 b9 53 5d 28 9e ..V31-5..00-5]6
00a0 10 f9 1c 71 0e 15 87 dc ab ab c5 a4 a0 60 8a 41 ..q.....A
00b0 93 c7 17 1b cb df 72 c6 8e 6f df b3 05 0f 16 7c .....P..g.....
00c0 08 fb 16 42 4e 36 33 34 c7 f5 09 4c 80 70 28 b4 ..BN634...L.p(
00d0 c4 08 c9 31 a3 4c 30 d0 25 e1 4b 35 2b 4e db 4c ...l.L0.%K5+N1

```

Frame (1238 bytes) Reassembled TCP (10725 bytes)

TCP Segment (tcp.segment), 1363 bytes

Packets: 29130 - Displayed: 34 (0.1%)

Profile: Default

9:47 03/10/2017

A su vez más abajo comprobamos como se vuelve a repetir el proceso de Hello y Hello answer a otra dirección.

Así hemos podido ver como se realizan las conexiones a diferentes IP's a la hora de buscar un recurso que tienen varias fuentes y como se realizan las peticiones de este

Aclaración de algunos paquetes y protocolo

En algunas imágenes podemos apreciar mensajes que en la info muestran: "Request Parts" esos son los mensajes en los que le pedimos a esta IP los paquetes.

Una vez descargado del todo mandamos un último aviso: Slot Release, dando una especie de "Adios y gracias amigo".

Una vez explicada la transferencia de paquetes, diré que todo se ha realizado por TCP, por la sencilla razón de que es una Imagen ISO, un archivo que necesita tener la consistencia al 100%, hasta el último bit de información y datos.

node.dat

Este archivo es una especie de libro de direcciones de los clientes. Y es necesario para tener conexión a dichos clientes. Es decir, si no tuviéramos este archivo o no estuviera conectado, no encontraríamos direcciones de las que descargar lo que buscamos.

Realizado por: J.Javier Galera Garrido