Práctica enrutamiento IP

Sumario

Introducción

La quinta práctica de planificación y administración de redes trata sobre la **capa de transporte**, es decir, la cuarta del modelo ISO/OSI empezando por el nivel físico. Este nivel es el primero que **une procesos** en vez de *hosts*, tal y como hacía el anterior, el de red. Suele ser la primera capa visible por los programadores, por lo que el nivel IP puede considerarse como la *frontera* entre estos y nosotros, los *administradores*.

La capa de transporte se compone de los protocolos **UDP** (*User Datagram Protocol*) y **TCP** (*Transmission Control Protocol*). El primero **no está orientado a conexión y no es fiable**, mientras que el segundo, TCP, **sí posee estas dos cualidades**.

El comando que emplearemos para poder escuchar en los puertos y conectarnos con otros ordenadores es `nc` (o más conocido como `netcat`), el cual puede manejar ambos protocolos según se especifica en su manual, pero el uso que se le dará mayoritariamente en este trabajo estará enfocado a TCP.

En el protocolo TCP, se establecen conexiones de la siguiente manera: Primero, **un servidor está escuchando en un puerto** y, posteriormente, **un cliente le envía una solicitud de conexión**. El servidor responde con una **aceptación de la conexión** y, después, el cliente ***acepta* la aceptación**, por lo que consigue conectarse. Hay diferentes estados:

- Cerrado: Ninguna conexión.
- Escuchando: Un servidor está esperando en un puerto a ser conectado.
- Establecido: Un cliente ha conectado con un servidor.
- Espera: Aguardando a que la conexión termine.

Los puertos, tanto de TCP como de UDP, son un total de 65536 ($\mathbf{2^{16}}$) y son asignados por la RFC 1700 de la siguiente manera:

- Puertos bien conocidos, que van del 0 al 1023 y son solo para el administrador.
- Puertos registrados, del 1024 al 49151, y son para servicios menos críticos.
- Puertos dinámicos, del 49152 al 65535, y son asignables a los clientes.

En Linux, la lista de puertos conocidos está en el fichero `/etc/services`.

El objetivo de esta práctica no es otro que –como en muchas otras prácticas– familiarizarse con los conceptos de esta unidad, como el de *puerto* o *conexión*. También, se trata de poder ser capaces de crear servicios simples de red, así como de utilizarlos y de monitorizar el estado de las conexiones de red.

Crear un chat con otros ordenadores del aula

Para crear un chat con otros ordenadores –en este caso no serán del aula, sino con una máquina virtual, pero los conceptos vienen a ser los mismos–, hay que ponerse a escuchar en un puerto con uno de los ordenadores y, con el otro, conectarte a él.

Para comprobar la información sobre los puertos, emplearemos el comando `ss` en vez del que se pide, `netstat`, ya que al menos en OpenSuse Tumbleweed 15.0, dicho comando está obsoleto, tal y como se indica en el manual:

```
This program is obsolete. Replacement for netstat is ss. Replacement for netstat -
r is ip route. Replacement for netstat -i is ip -s link. Replacement for netstat -
g is ip maddr.
```

Por esta razón, emplearemos el comando `ss`, con las mismas opciones que con `netstat`, o séase, `-antp`, ya que son las mismas, según lo especifica su manual:

```
       -a, --all
              Display both listening and non-listening (for TCP this means
              established connections) sockets.

       -n, --numeric
              Do not try to resolve service names.

       -p, --processes
              Show process using socket.

       -t, --tcp
              Display TCP sockets.
```

Para la realización de este apartado, escucharemos en un puerto en el ordenador `linux-ewxc` –por lo tanto, será el servidor– y comprobaremos los puertos en los que estamos escuchando con el comando `ss`, aunque antes de todo comprobaremos qué IP tenemos:

```
linux-ewxc:/home/lg-gram # ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: wlp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group
default qlen 1000
    link/ether a0:c5:89:e1:4f:1f brd ff:ff:ff:ff:ff:ff
```

```
    inet 192.168.1.134/24 brd 192.168.1.255 scope global dynamic noprefixroute
wlp2s0
       valid_lft 258540sec preferred_lft 258540sec
    inet6 fe80::dadb:5854:83dd:2a7e/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
linux-ewxc:/home/lg-gram # nc -l 60093
```

```
linux-ewxc:/home/lg-gram # ss -antp | grep State && ss -antp | grep 60093
State  Recv-Q Send-Q          Local Address:Port             Peer Address:Port
LISTEN 0      1                     0.0.0.0:60093                 0.0.0.0:*
users:(("nc",pid=9363,fd=3))
```

```
[0] 0:bash*                                          "linux-ewxc" 16:09 09-mar-19
```

Ahora, nos conectaremos desde `alumno` –que será el cliente– a `linux-ewxc`, mediante el puerto `60093`. También, comprobaremos las conexiones establecidas en dicho puerto:

```
alumno:/home/alumno # nc 192.168.1.134 60093
```

```
alumno:/home/alumno # ss -antp | grep State && ss -antp | grep 60093
State  Recv-Q Send-Q          Local Address:Port              Peer Address:Port
ESTAB  0      0               192.168.1.136:50584           192.168.1.134:60093
users:(("nc",pid=3243,fd=3))
```

```
[0] 0:bash*                                          "alumno" 16:11 09-mar-19
```

Procedemos a hacerlo al revés, siendo `alumno` el cliente y `linux-ewxc` el servidor.

Comprobamos qué IP tenemos con `alumno`, escuchamos en un puerto y comprobamos los puertos en los que estamos escuchando:

```
alumno:/home/alumno # ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 08:00:27:75:00:ff brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.136/24 brd 192.168.1.255 scope global dynamic noprefixroute
enp0s3
```

```
       valid_lft 258999sec preferred_lft 258999sec
    inet6 fe80::f3f6:6025:c3e7:4ae8/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
alumno:/home/alumno # nc -l 60093
```

```
alumno:/home/alumno # ss -antp | grep State && ss -antp | grep 60093
State  Recv-Q Send-Q          Local Address:Port             Peer Address:Port
LISTEN 0      1                     0.0.0.0:60093                 0.0.0.0:*
users:(("nc",pid=3898,fd=3))
```

```
[0] 0:bash*                                              "alumno" 16:12 09-mar-19
```

Nos conectamos desde `linux-ewxc` a `alumno` mediante el puerto `60093` y comprobamos las conexiones establecidas en dicho puerto:

```
linux-ewxc:/home/lg-gram # nc 192.168.1.136 60093
```

```
linux-ewxc:/home/lg-gram # ss -antp | grep State && ss -antp | grep 60093
State  Recv-Q Send-Q          Local Address:Port             Peer Address:Port
ESTAB     0      0               192.168.1.134:60032
192.168.1.136:60093                  users:(("nc",pid=9623,fd=3))
```

```
[0] 0:bash*                                            "linux-ewxc" 16:13 09-mar-19
```

Finalmente, nos conectaremos con dos compañeros de clase, por lo que escucharemos en puertos diferentes, para evitar el conflicto producido al escuchar en la misma IP y en el mismo puerto simultáneamente.

Escuchamos con `linux-ewxc` en dos puertos distintos:

```
linux-ewxc:/home/lg-gram # nc -l 60094
```

```
linux-ewxc:/home/lg-gram # nc -l 60093
```

```
[0] 0:bash*                                            "linux-ewxc" 16:28 09-mar-19
```

Nos conectamos desde `alumno` a `linux-ewxc` por el puerto `60094`:

```
alumno:/home/alumno # nc 192.168.1.134 60094
```

```
alumno:/home/alumno # ss -antp | grep State && ss -antp | grep 60094
State  Recv-Q Send-Q          Local Address:Port            Peer Address:Port
ESTAB        0        0          192.168.1.136:57970      192.168.1.134:60094
users:(("nc",pid=5873,fd=3))
```

```
[0] 0:bash*                                              "alumno" 16:29 09-mar-19
```

Y también nos conectamos desde `otro` a `linux-ewxc` por el puerto `60093`:

```
otro:/home/otro # nc 192.168.1.134 60093
```

```
otro:/home/otro # ss -antp | grep State && ss -antp | grep 60093
State  Recv-Q Send-Q          Local Address:Port            Peer Address:Port
ESTAB        0        0          192.168.1.137:47062      192.168.1.134:60093
users:(("nc",pid=2974,fd=3))
```

```
[0] 0:bash*                                               "otro" 16:30 09-mar-19
```

Por último, comprobamos las conexiones de `linux-ewxc`:

```
linux-ewxc:/home/lg-gram # nc -l 60094
```

```
linux-ewxc:/home/lg-gram # nc -l 60093
```

```
linux-ewxc:/home/lg-gram # ss -antp | grep State && ss -antp | grep 6009
State  Recv-Q Send-Q          Local Address:Port            Peer Address:Port
LISTEN     1      1                    0.0.0.0:60093                0.0.0.0:*
users:(("nc",pid=9929,fd=3))
LISTEN     0      1                    0.0.0.0:60094                0.0.0.0:*
users:(("nc",pid=10040,fd=3))
ESTAB      0      0               192.168.1.134:60093
192.168.1.137:47062                users:(("nc",pid=9929,fd=4))
ESTAB      0      0               192.168.1.134:60094
192.168.1.136:57970                users:(("nc",pid=10040,fd=4))
```

```
[0] 0:bash*                                          "linux-ewxc" 16:30 09-mar-19
```

Proteger a un servidor de chat

Crearemos un servidor en `linux-ewxc` que escuche en la dirección IP de nuestro ordenador y otro en la dirección `127.0.0.1`, ambos en el puerto `60093`:

```
linux-ewxc:/home/lg-gram # nc -l 192.168.1.134 60093

linux-ewxc:/home/lg-gram # nc -l 127.0.0.1 60093

[0] 0:bash*                                          "linux-ewxc" 18:12 14-mar-19
```

Estos pueden funcionar simultáneamente en el mismo puerto debido a que la clave primaria de escucha en un puerto es la dirección IP más el puerto, a la vez, por lo que el identificador no es el mismo, ya que sus direcciones IP son distintas.

El usuario `alumno` intentará conectarse por ambas direcciones IP y puertos:

```
alumno:/home/alumno # nc 192.168.1.134 60093

alumno:/home/alumno # nc 127.0.0.1 60093
alumno:/home/alumno #

[0] 0:bash*                                          "alumno" 18:10 14-mar-19
```

Con la primera dirección, la `192.168.1.134` sí logra conectarse, pero al intentarlo por la `127.0.0.1` no puede debido a que dicha dirección hace referencia a uno mismo y este se cree que es él, y ese puerto no lo hemos abierto nosotros manualmente.

Por el contrario, `linux-ewxc` sí se puede conectar a dicha IP por ese puerto:

```
linux-ewxc:/home/lg-gram # nc -l 192.168.1.134 60093

linux-ewxc:/home/lg-gram # nc -l 127.0.0.1 60093

linux-ewxc:/home/lg-gram # nc 127.0.0.1 60093

linux-ewxc:/home/lg-gram # ss -antp | grep State && ss -antp | grep 60093
State  Recv-Q Send-Q          Local Address:Port             Peer Address:Port
LISTEN 0      1                   127.0.0.1:60093                 0.0.0.0:*
users:(("nc",pid=11120,fd=3))
LISTEN 0      1               192.168.1.134:60093                 0.0.0.0:*
users:(("nc",pid=11118,fd=3))
ESTAB  0      0                   127.0.0.1:60093              127.0.0.1:54948
users:(("nc",pid=11120,fd=4))
ESTAB  0      0                   127.0.0.1:54948              127.0.0.1:60093
users:(("nc",pid=11295,fd=3))
ESTAB  0      0               192.168.1.134:60093          192.168.1.136:45482
users:(("nc",pid=11118,fd=4))

[0] 0:bash*                                                "linux-ewxc" 18:12 14-mar-19
```

Enviar información de ficheros

Vamos a enviar información de los directorios de $HOME por medio de tuberías.

Primero, nos ponemos a escuchar en alumno en el puerto 60093:

```
alumno:/home/alumno # nc -l 60093
```

Después, enviamos la información los directorios de nuestra $HOME de linux-ewxc a alumno:

```
linux-ewxc:/home/lg-gram # ls -la $HOME | nc 192.168.1.136 60093
```

Y vemos la información que ha recibido alumno de linux-ewxc:

```
alumno:/home/alumno # nc -l 60093
total 72
drwx------ 1 root root   322 mar  9 16:38 .
drwxr-xr-x 1 root root   172 feb 23 11:54 ..
-rw------- 1 root root 13492 mar  9 16:38 .bash_history
-rw-r--r-- 1 root root   156 ene 23 12:35 .bashrc
drwxr-xr-x 1 root root     0 ene 30 00:30 bin
drwx------ 1 root root   194 feb 21 23:43 .cache
drwxr-xr-x 1 root root    22 feb  1 15:53 .config
drwx------ 1 root root    22 ene 23 22:20 .dbus
drwx------ 1 root root    22 ene 30 00:30 .gnupg
drwxr-xr-x 1 root root    36 ene 23 11:59 inst-sys
-rw------- 1 root root   138 mar  4 06:21 .lesshst
-rw------- 1 root root    39 feb 13 12:46 .node_repl_history
drwxr-xr-x 1 root root     8 ene 23 13:02 oradiag_root
-rw------- 1 root root 14262 mar  8 09:36 .sqlplus_history
drwx------ 1 root root    22 ene 26 20:16 .ssh
-rw------- 1 root root 13906 mar  6 10:02 .viminfo
drwx------ 1 root root     0 feb 24 15:19 .w3m
-rw------- 1 root root    55 mar  9 16:02 .xauthkaSgpS
-rw------- 1 root root    55 mar  9 16:35 .xauthU38B1X
-rw------- 1 root root    55 mar  9 15:58 .xauthu6eBiH
```

Invertimos el papel de cliente y servidor para enviar la ayuda del comando nc.

Nos ponemos en modo de escucha en `linux-ewxc`:

```
linux-ewxc:/home/lg-gram # nc -l 60093
```

Enviamos la información del comando nc de `alumno` a `linux-ewxc`:

```
alumno:/home/alumno # man nc | nc 192.168.1.134 60093
```

En `linux-ewxc` vemos cómo hemos recibido el manual del comando nc:

```
linux-ewxc:/home/lg-gram # nc -l 60093
           $ nc -x10.2.3.4:8080 -Xconnect host.example.com 42

     The same example again, this time enabling proxy authentication with username
“ruser” if the proxy requires
     it:

           $ nc -x10.2.3.4:8080 -Xconnect -Pruser host.example.com 42

SEE ALSO
     cat(1), ssh(1)

AUTHORS
     Original implementation by *Hobbit* <hobbit@avian.org>.
     Rewritten with IPv6 support by
     Eric Jackson <ericj@monkey.org>.
     Modified for Debian port by Aron Xu ⟨aron@debian.org⟩.

CAVEATS
     UDP port scans using the -uz combination of flags will always report success
irrespective of the target ma-
     chine's state.  However, in conjunction with a traffic sniffer either on the
target machine or an intermedi-
     ary device, the -uz combination could be useful for communications
diagnostics.  Note that the amount of UDP
     traffic generated may be limited either due to hardware resources and/or
configuration settings.

BSD                                                 September 25, 2018
BSD
```

De la misma manera, solo que grabando la información recibida a un archivo, transferiremos una imagen al otro y viceversa:

Nos ponemos en modo de escucha en `alumno` en el puerto `60093` y nos preparamos para grabar la información que llegue a un archivo de imagen:

```
alumno:/home/alumno # nc -l 60093 > imagen.jpg
```

Enviamos la foto de `linux-ewxc` a `alumno` y comprobamos cuánto ocupa esta:

```
linux-ewxc:/home/lg-gram # cat opensuse-1551910063720-5495.jpg | nc 192.168.1.136
60093
```

```
linux-ewxc:/home/lg-gram # ls -l opensuse-1551910063720-5495.jpg
-rw-r--r-- 1 root root 135470 mar  9 16:40 opensuse-1551910063720-5495.jpg

[0] 0:bash*                                          "linux-ewxc" 16:49 09-mar-19
```

Recibimos la imagen en `alumno` y comprobamos su peso para verificar que se ha transmitido todo el contenido. Ejecutamos el comando de visualización de la misma:

```
alumno:/home/alumno # nc -l 60093 > imagen.jpg
alumno:/home/alumno #
```

```
alumno:/home/alumno # ls -l imagen.jpg
-rw-r--r-- 1 root root 135470 mar  9 16:47 imagen.jpg
alumno:/home/alumno # display imagen.jpg

[0] 0:bash*                                            "alumno" 16:48 09-mar-19
```

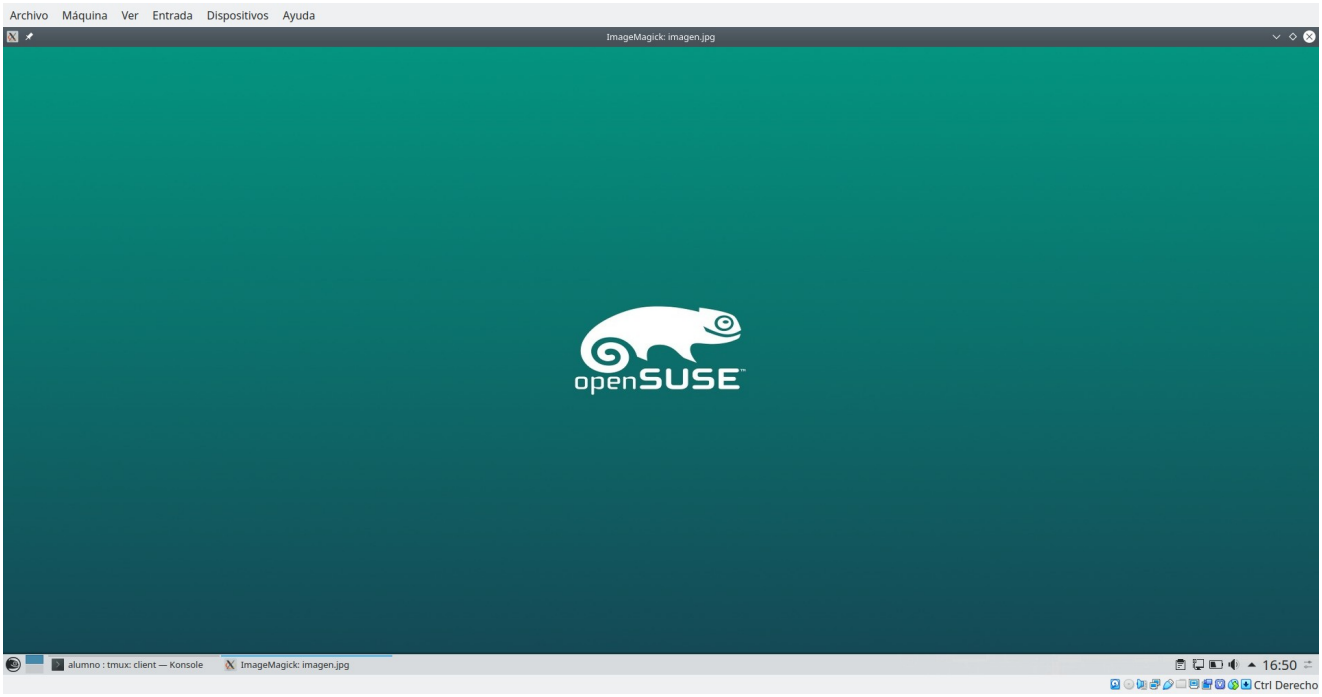

***Figura 1**: Visualización de la imagen recibida por `alumno`*

Cambiamos los papeles de servidor y cliente.

Entramos en modo de escucha con `linux-ewxc` en el puerto `60093`:

```
linux-ewxc:/home/lg-gram # nc -l 60093 > imagen.jpg
```

Enviamos la foto de `alumno` a `linux-ewxc` y comprobamos su peso

```
alumno:/home/alumno # cat imagen.jpg | nc 192.168.1.134 60093

─────────────────────────────────────────────────────────────────────────────

alumno:/home/alumno # ls -l imagen.jpg
-rw-r--r-- 1 root root 135470 mar  9 16:47 imagen.jpg

[0] 0:bash*                                                  "alumno" 16:51 09-mar-19
```

Recibimos la imagen en `linux-ewxc` –hay veces hay que cortar manualmente la transferencia de un archivo, antes revisamos que tengan el mismo tamaño– y comprobamos cuánto pesa para verificar que se ha transmitido todo el contenido. Ejecutamos el comando de visualización de la misma:

```
linux-ewxc:/home/lg-gram # nc -l 60093 > imagen.jpg
^C
linux-ewxc:/home/lg-gram #
```

```
linux-ewxc:/home/lg-gram # ls -l imagen.jpg
-rw-r--r-- 1 root root 135470 mar  9 16:52 imagen.jpg
linux-ewxc:/home/lg-gram # display imagen.jpg
```

```
[0] 0:bash*                                          "linux-ewxc" 16:53 09-mar-19
```

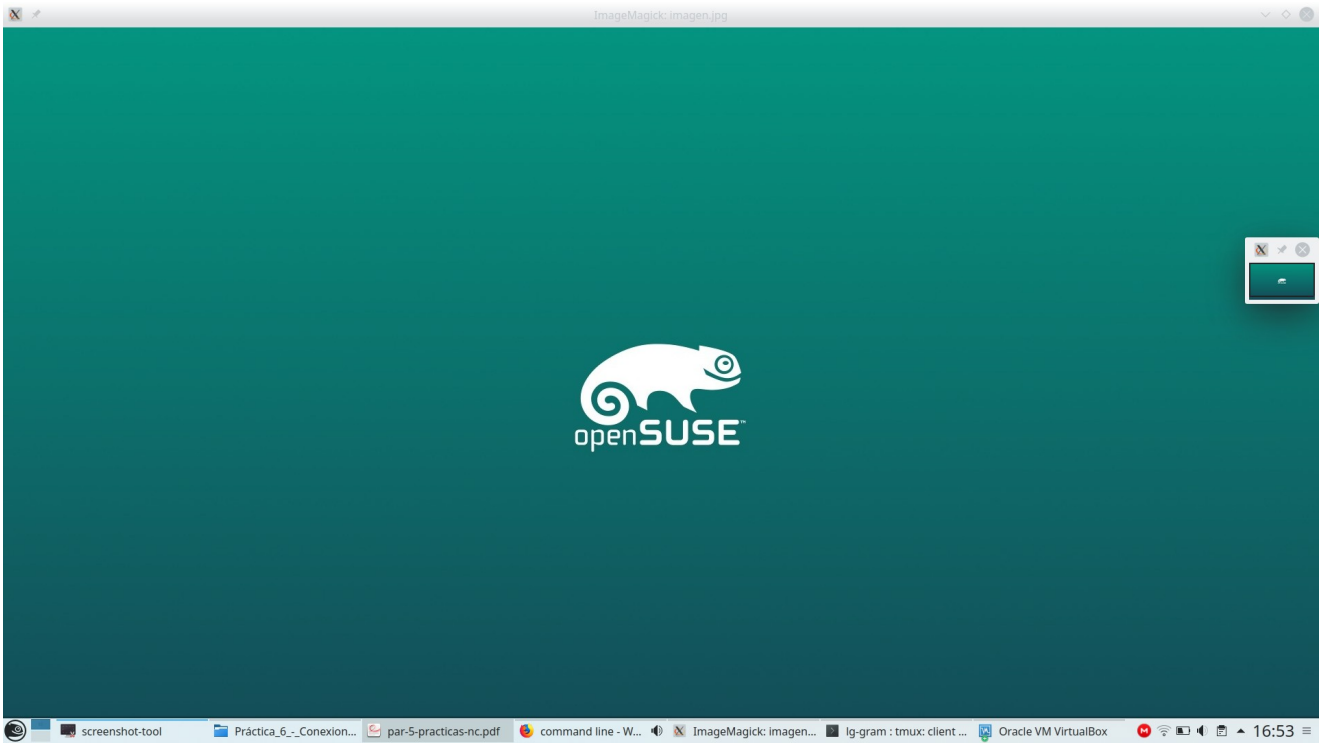

***Figura 2**: Visualización de la imagen recibida por `linux-ewxc`*

Enviar una máquina virtual a un compañero

Con una máquina virtual en un fichero `.iso`, `.ova`, etc., vamos a transferir una a algún compañero –o máquina virtual–, midiendo el tiempo de transmisión con el comando `time` y, en el receptor, monitorizando el progreso del directorio de destino con `watch`.

Entramos al modo de escucha desde `alumno` en el puerto `60093` y nos preparamos para grabar la información que llegue a un archivo `.ova`:

```
alumno:/home/alumno # nc -l 60093 > maquina.ova
```

Enviamos la máquina virtual de `linux-ewxc` a `alumno` con la opción de medir el tiempo para el siguiente comando que sea ejecutado:

```
linux-ewxc:/home/lg-gram # time cat OpenSolaris2009.ova | nc 192.168.1.136 60093
```

En `alumno` vemos cómo va aumentando el tamaño de nuestro archivo:

```
alumno:/home/alumno # nc -l 60093 > maquina.ova
```

```
alumno:/home/alumno # watch -d ls -l
Every 2,0s: ls -l                          192.168.1.136: Sat Mar  9 17:03:26 2019

total 693880
drwxr-xr-x 2 alumno users       4096 mar  4 13:48 bin
drwxr-xr-x 2 alumno users       4096 mar  4 13:50 Descargas
drwxr-xr-x 2 alumno users       4096 mar  4 13:50 Documentos
drwxr-xr-x 2 alumno users       4096 mar  4 13:51 Escritorio
drwxr-xr-x 2 alumno users       4096 mar  4 13:50 Imágenes
-rw-r--r-- 1 root   root      135470 mar  9 16:47 imagen.jpg
-rw-r--r-- 1 root   root   710335712 mar  9 17:03 maquina.ova
drwxr-xr-x 2 alumno users       4096 mar  4 13:50 Música
drwxr-xr-x 2 alumno users       4096 mar  4 13:50 Plantillas
drwxr-xr-x 2 alumno users       4096 mar  4 13:50 Público
-rw-r--r-- 1 root   root          0 mar  9 16:06 redes
-rw-r--r-- 1 root   root       1288 mar  9 16:04 redes1-2.txt
-rw-r--r-- 1 root   root       1288 mar  9 16:10 redes2-2.txt
-rw-r--r-- 1 root   root       1288 mar  9 16:27 redes3-2.txt
-rw-r--r-- 1 root   root       1127 mar  9 16:35 redes4-2.txt
drwxr-xr-x 2 alumno users       4096 mar  4 13:50 Vídeos

[0] 0:watch*                                              "alumno" 17:03 09-mar-19
```

El archivo ya ha sido transferido a `alumno`, por eso el comando `watch` ya no cambia:

```
alumno:/home/alumno # nc -l 60093 > maquina.ova
```

```
alumno:/home/alumno # watch -d ls -l
Every 2,0s: ls -l                                   alumno: Sat Mar  9 17:04:03 2019

total 1407540
drwxr-xr-x 2 alumno users       4096 mar  4 13:48 bin
drwxr-xr-x 2 alumno users       4096 mar  4 13:50 Descargas
drwxr-xr-x 2 alumno users       4096 mar  4 13:50 Documentos
drwxr-xr-x 2 alumno users       4096 mar  4 13:51 Escritorio
drwxr-xr-x 2 alumno users       4096 mar  4 13:50 Imágenes
-rw-r--r-- 1 root   root      135470 mar  9 16:47 imagen.jpg
-rw-r--r-- 1 root   root  1441121280 mar  9 17:03 maquina.ova
drwxr-xr-x 2 alumno users       4096 mar  4 13:50 Música
drwxr-xr-x 2 alumno users       4096 mar  4 13:50 Plantillas
drwxr-xr-x 2 alumno users       4096 mar  4 13:50 Público
-rw-r--r-- 1 root   root           0 mar  9 16:06 redes
-rw-r--r-- 1 root   root        1288 mar  9 16:04 redes1-2.txt
-rw-r--r-- 1 root   root        1288 mar  9 16:10 redes2-2.txt
-rw-r--r-- 1 root   root        1288 mar  9 16:27 redes3-2.txt
-rw-r--r-- 1 root   root        1127 mar  9 16:35 redes4-2.txt
drwxr-xr-x 2 alumno users       4096 mar  4 13:50 Vídeos


[0] 0:watch*                                                  "alumno" 17:04 09-mar-19
```

Volvemos a `linux-ewxc` para revisar cuánto tiempo ha tardado en total la transferencia, el cual es mayor al tiempo que ha durado realmente, ya que lo hemos tenido que cortar manualmente:

```
linux-ewxc:/home/lg-gram # time cat OpenSolaris2009.ova | nc 192.168.1.136 60093
^C

real    0m41,406s
user    0m0,446s
sys     0m5,072s
```

Volvemos a hacer la misma serie de operaciones, de manera inversa:

Entramos al modo de escucha desde `linux-ewxc` en el puerto `60093` y nos preparamos para grabar la información que llegue a un archivo `.ova`:

```
linux-ewxc:/home/lg-gram # nc -l 60093 > 0maquina.ova
```

Enviamos la máquina virtual de `alumno` a `linux-ewxc` con la opción de medir el tiempo para el siguiente comando que sea ejecutado:

```
alumno:/home/alumno # time cat maquina.ova | nc 192.168.1.134 60093
```

En `linux-ewxc` vemos cómo va aumentando el tamaño de nuestro archivo:

```
linux-ewxc:/home/lg-gram # nc -l 60093 > 0maquina.ova
```

```
linux-ewxc:/home/lg-gram # watch -d ls -l
Every 2,0s: ls -l                          linux-ewxc: Sat Mar  9 17:13:08 2019

total 1932812
-rw-r--r-- 1 root    root   532905984 mar  9 17:13 0maquina.ova
-rw-r--r-- 1 lg-gram users     408193 mar  9 16:11 1.png
-rw-r--r-- 1 lg-gram users     412856 mar  9 16:13 2.png
-rwxr-xr-x 1 lg-gram users         68 feb  6 23:13 academia.sql
drwxr-xr-x 1 lg-gram users          0 ene 23 12:18 bin
-rwxr-xr-x 1 lg-gram users       2159 feb  9 23:41 Ciudades_España.txt
-rwxr-xr-x 1 lg-gram users      88192 ene 30 13:46 datamodeler.log
drwxr-xr-x 1 lg-gram users        226 mar  9 14:42 Descargas
drwxr-xr-x 1 lg-gram users       2024 mar  6 23:30 Documentos
drwxr-xr-x 1 lg-gram users       1314 mar  4 11:49 Escritorio
drwxr-xr-x 1 lg-gram users         82 mar  8 15:43 Imágenes

[0] 0:watch*                                          "linux-ewxc" 17:13 09-mar-19
```

El archivo ya ha sido transferido a `linux-ewxc`:

```
linux-ewxc:/home/lg-gram # nc -l 60093 > 0maquina.ova
```

```
linux-ewxc:/home/lg-gram # watch -d ls -l
Every 2,0s: ls -l                          linux-ewxc: Sat Mar  9 17:13:08 2019

total 2820264
-rw-r--r-- 1 root    root  1441121280 mar  9 17:13 0maquina.ova
-rw-r--r-- 1 lg-gram users     408193 mar  9 16:11 1.png
-rw-r--r-- 1 lg-gram users     412856 mar  9 16:13 2.png
-rwxr-xr-x 1 lg-gram users         68 feb  6 23:13 academia.sql
drwxr-xr-x 1 lg-gram users          0 ene 23 12:18 bin
-rwxr-xr-x 1 lg-gram users       2159 feb  9 23:41 Ciudades_España.txt
-rwxr-xr-x 1 lg-gram users      88192 ene 30 13:46 datamodeler.log
drwxr-xr-x 1 lg-gram users        226 mar  9 14:42 Descargas
drwxr-xr-x 1 lg-gram users       2024 mar  6 23:30 Documentos
drwxr-xr-x 1 lg-gram users       1314 mar  4 11:49 Escritorio
drwxr-xr-x 1 lg-gram users         82 mar  8 15:43 Imágenes

[0] 0:watch*                                          "linux-ewxc" 17:13 09-mar-19
```

Volvemos a `alumno` para revisar cuánto tiempo ha tardado en total la transferencia y, como antes, hemos tenido que finalizar la conexión manualmente:

```
linux-ewxc:/home/lg-gram # time cat maquina.ova | nc 192.168.1.134 60093
^C

real    0m27,647s
user    0m0,078s
sys     0m16,201s
```

Simulación de un servidor web

Crearemos un servidor en el puerto 60093 y, utilizando un navegador web, visitaremos el servidor en la URL http://localhost:60093. Hay que conseguir que en este se vea representado, mediante un archivo HTML o introducido a mano el código, un título de nivel 1 (<h1>), letra negrita (<b>) y cursiva (<i>). Tiene que funcionar tanto en Mozilla Firefox como en un navegador de la rama de Google Chrome, es decir, *blink-based*.

Para esto, hay que emplear un lenguaje que los navegadores entiendan. Después de ponernos a escuchar en el puerto 60093, una vez el navegador haya introducido la anterior URL, automáticamente en el lado del servidor aparecerán las siguientes líneas, en azul:

```
linux-ewxc:/home/lg-gram # nc -l 60093
```

```
GET / HTTP/1.1
Host: localhost:60093
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

El lenguaje que entienden se encabeza por las siguientes tres primeras líneas, pudiendo la codificación (charset) incrustarse directamente en el HTML. Obligatoriamente han de realizarse dos saltos de línea entre content y el principio del lenguaje HTML:

```
HTTP/1.1
Server:Hola
Content-Type:text/html;charset=UTF-8


<!DOCTYPE html>
<html>
<head>
<title>
Servidor web
</title>
</head>
<body>
<h1>
Un título de nivel 1
</h1>
<p>
Un ejemplo de página que tiene
<b>
negrita
</b>
 y
<i>
```

```
cursiva.
</i>
</body>
</html>
```

Y se vería representado en nuestro navegador de la siguiente manera:

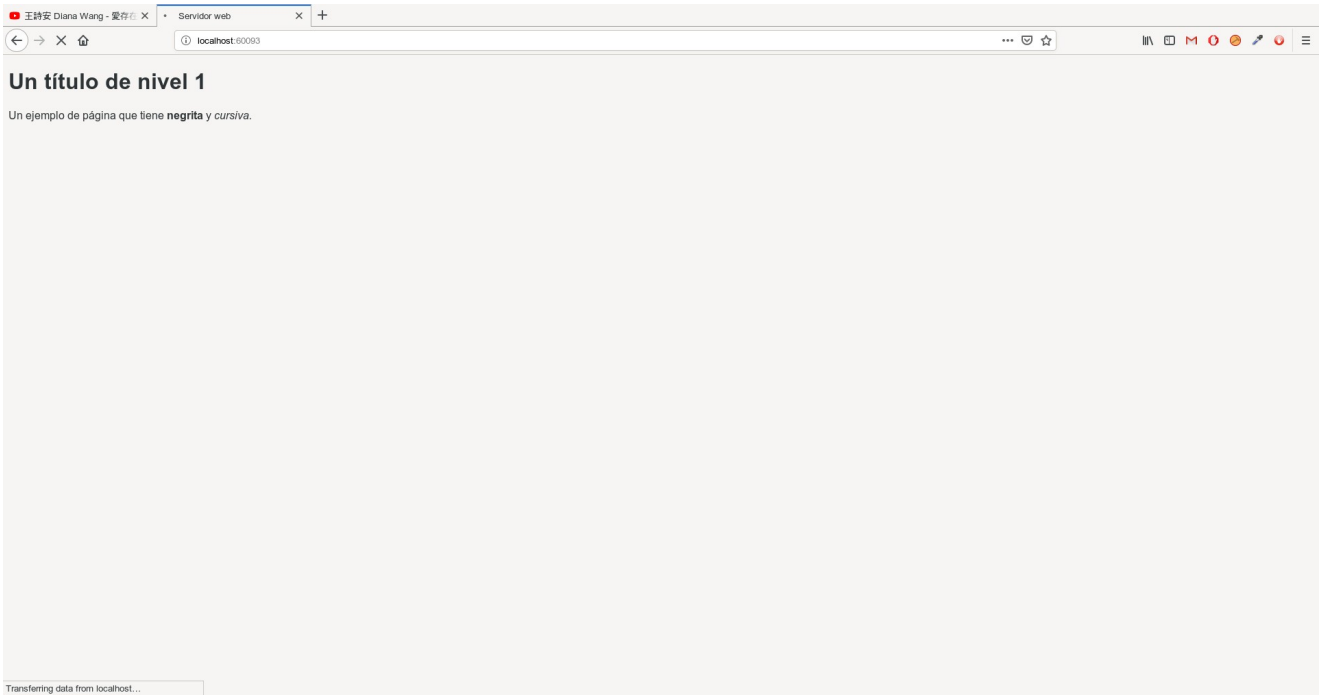

***Figura 3**: Vista de nuestra página web creada en el navegador Mozilla Firefox*

Si repetimos los pases anteriores pero ahora con un navegador que derive de Chromium, como Google Chrome o Vivaldi en este caso, las etiquetas que hemos puesto se verán únicamente como código:

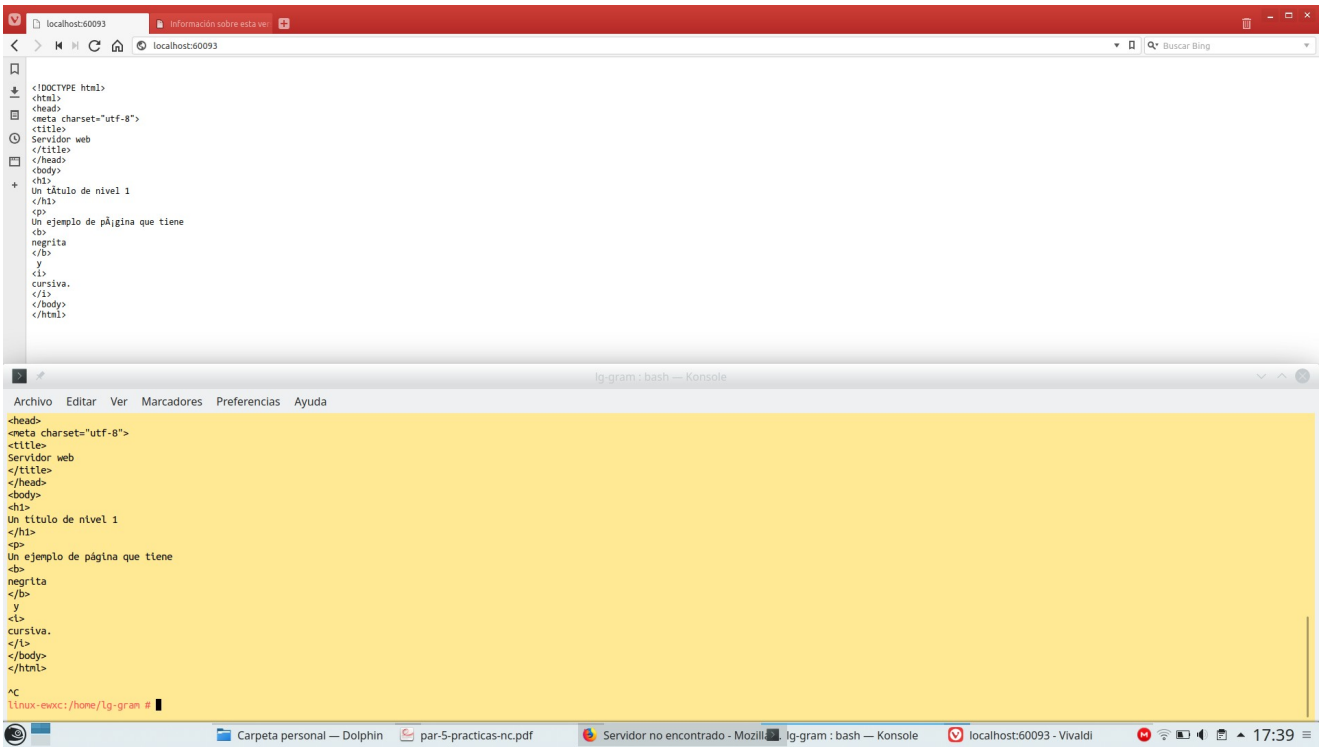

***Figura 4**: Código anterior ahora en Vivaldi. Solo se ven las etiquetas no se ven, estas no adquieren una forma*

Esto es debido a que este tipo de navegadores requieren de algo más para que funcionen, concretamente de una pequeña confirmación por parte del servidor en el encabezamiento, por lo que simplemente añadiremos `200 OK` después de `HTTP/1.1`:

```
linux-ewxc:/home/lg-gram # nc -l 60093
GET / HTTP/1.1
Host: localhost:60093
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/70.0.3538.113 Safari/537.36 Vivaldi/2.1.1337.51
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/
apng,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: es-ES,es;q=0.9

HTTP/1.1 200 OK
Server:Hola
Content-Type:text/html;charset=UTF-8



<!DOCTYPE html>
```

```
<html>
<head>
<title>
Servidor web
</title>
</head>
<body>
<h1>
Un título de nivel 1
</h1>
<p>
Un ejemplo de página que tiene
<b>
negrita
</b>
 y
<i>
cursiva.
</i>
</body>
</html>
```

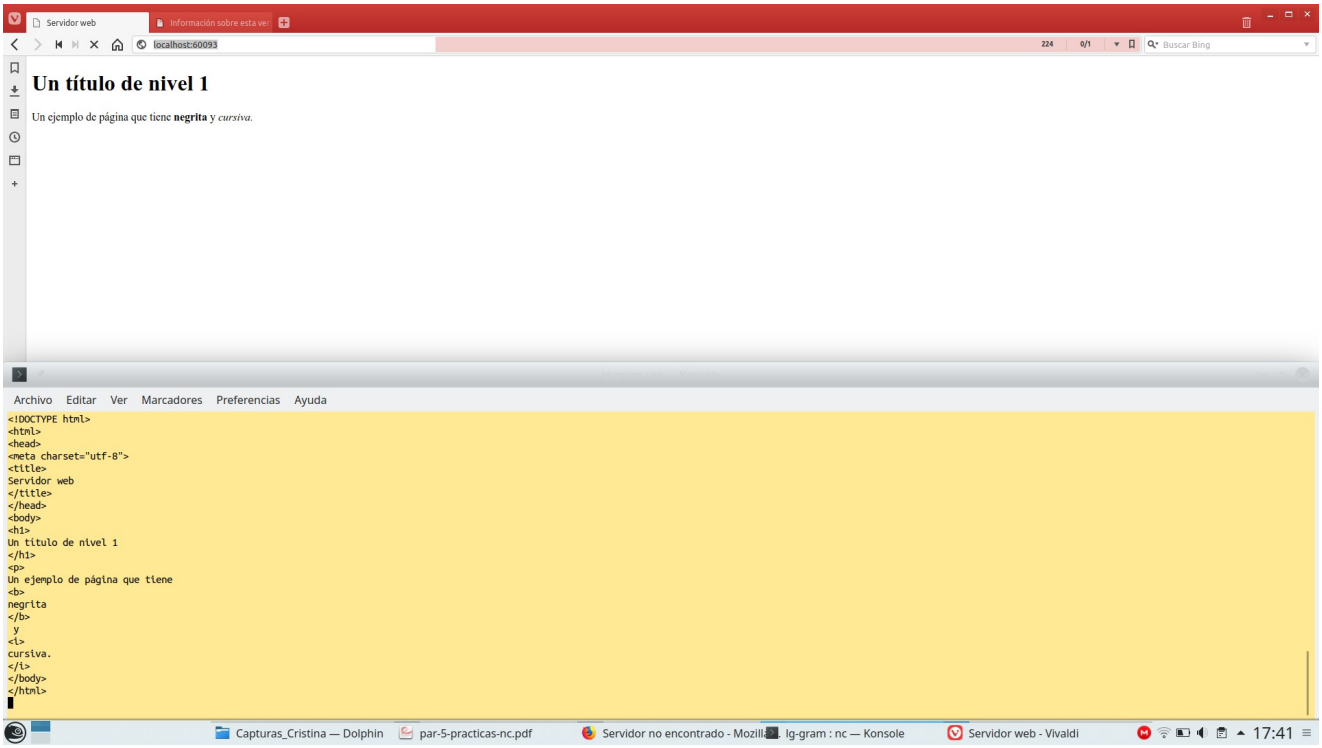

***Figura 5**: En Vivaldi se requiere de la confirmación `200 OK` por parte del servidor para que funcione correctamente*

Simulación de un cliente web

Ya para acabar, utilizaremos directamente el protocolo HTTP para pedir una página a un servidor, el de la DGT, para ver si hay atasco o no de camino a casa. El puerto del protocolo HTTP es el 80, y la dirección web de la DGT es www.dgt.es. Con las dos primeras líneas y después los dos saltos de línea, solicitaremos el archivo deseado:

```
linux-ewxc:/home/lg-gram # nc www.dgt.es 80
GET /gsm.txt HTTP/1.1
Host:www.dgt.es


HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: max-age=120
Content-Type: text/plain; charset=UTF-8
Date: Sat, 09 Mar 2019 16:41:42 GMT
Etag: "6e7fa-693-583abfdcb7877"
Expires: Sat, 09 Mar 2019 16:43:42 GMT
Last-Modified: Sat, 09 Mar 2019 16:40:04 GMT
Server: ECAcc (toj/8918)
X-Cache: HIT
X-UA-Compatible: IE=edge
Content-Length: 1683

CLUMONDO◆EDO                              VVA-8        536 552*
CNAIRATI                                  NNNA-2012      8   8*
CNAPIKATUA                                NNNA-2011      7  11*
RGRSIERRA NEVADA                          1NA-4025       0   8*
RSACANDELARIO                             1NDSA-191      8  11*
RTFBUENAVISTA DEL NORTE                   1NTF-445       3   9*
OBISANTURTZI                               NN-644      130 130*SANTURTZI
OGICABANELLES                              NN-260       53  57*DESVIAMENT PER N-
260A
OMASIERRA DE YEGUAS                        NA-7279       0  15*
OMAVI◆UELA                                 NA-7205      11  11*
ONATUDELA                                  NN-121C       2   2*
RB BARCELONA                              2AB-20         6   9SNUS LLOBREGAT-UN
CARRIL TALLAT
RB BARCELONA                              1AB-20         6   5NNUS TRINITAT
RB CASTELLBELL I EL VILAR                 1NC-1411A     12  13*DE 15:30 A 18
HORES
RB MOI◆                                   5NC-59        38  39*RUA DE
CARNESTOLTES DE MOI◆
RCASAN FERNANDO                           3ACA-33       13  13-C◆DIZ
$$
```

Pues parece que por Alcalá de Henares no hay atasco hoy, ¡así que llegaremos a nuestra hora a casa!