UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA
UNAN-LEON

FACULTAD DE CIENCIA Y TECNOLOGIA

INGENIERIA EN TELEMATICA

Practica 6

Asignatura:
Redes de Computadoras

Integrante:
- Bismarck Antonio Berrios

1. Comunicación de aplicaciones usando el protocolo UDP

1.1. Análisis de captura de tráfico UDP

En la captura udp.cap se muestra una comunicación UDP. Contesta a las siguientes preguntas:

1. Cuáles son las direcciones IP y puertos involucrados en la comunicación?
   10.0.0.1, 10.0.0.2 y 11.0.0.2 y el puerto 17

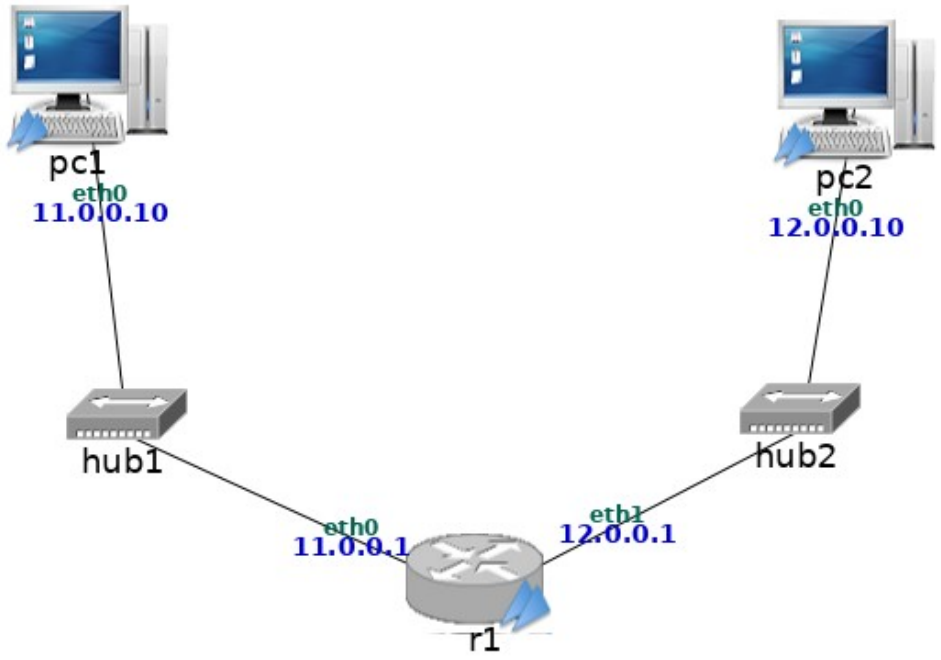
2. Qué información puedes extraer de la captura sobre la red en la que se ha realizado la captura?
   Que la comunicación pasa atravez de un router

3. Cuál es el número de paquetes UDP y número de bytes de datos intercambiados?
   2 paquetes UDP que suman 11 bytes de datos en total.

1.2. Estudio de UDP mediante aplicaciones cliente y servidor lanzadas con nc

Descarga de la página de la asignatura el fichero lab-p5.tgz , que contiene un escenario de red. Descomprímelo de la misma manera que hiciste en prácticas anteriores.

Lanza ahora NetGUI. En el menú, elige File → Open y selecciona la carpeta lab-p5 en la que está el escenario. Verás aparecer la red de la figura 1



Arranca las máquinas de una en una, esperando que una máquina haya terminado su arranque antes de arrancar la siguiente.

En este apartado utilizarás la orden nc para observar el funcionamiento de UDP en diversas situaciones. Ve ahora al anexo de esta práctica en el que se explica cómo utilizar nc para arrancar clientes y servidores UDP, y vuelve aquí después para continuar.

1.2.1. UDP es un protocolo basado en datagramas: no hay establecimiento de conexión

1. Inicia una captura en el router r1 1 .

```
r1:~# tcpdump -i eth0 -s 0 -w /hosthome/r1.cap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 byte
s
```

2. Usando nc lanza una aplicación servidor UDP en la máquina 12.0.0.10 y puerto 11111: nc -u -l 11111

```
pc2:~# nc -u -l -p 11111
```

3. Usando nc lanza una aplicación cliente UDP en la máquina 11.0.0.10 para que se comunique con el servidor (no envíes datos ni desde el cliente al servidor ni desde el servidor al cliente), desde el puerto local 33333: nc -u -p 33333 12.0.0.10 11111

```
pc1:~# nc -u -p 33333 12.0.0.10 11111
```

4. Interrumpe la captura.

Explica qué paquetes deberían haberse capturado. Observa la captura y comprueba tu suposición.

No debería de haberse capturado nada debido a que no se envió ningún paquete.

1.2.2. Fragmentación IP con envíos UDP

1. Inicia una nueva captura en el router r1 para que guarde los paquetes capturados en un fichero.

```
r1:~# tcpdump -i eth0 -s 0 -w /hosthome/1.2.2-r1.cap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 byte
s
```

2. Escribe en el terminal donde tienes lanzado el cliente 20 líneas de texto, pulsando una letra cualquiera del teclado (con el tamaño por defecto del terminal de NetGUI, cada línea permite escribir 80 caracteres, así que estarás generando una línea de 80x20=1600 carácteres, cada uno de ellos ocupando 1 byte).

3. A continuación pulsa la tecla INTRO o ENTER (véase la #gura 2).

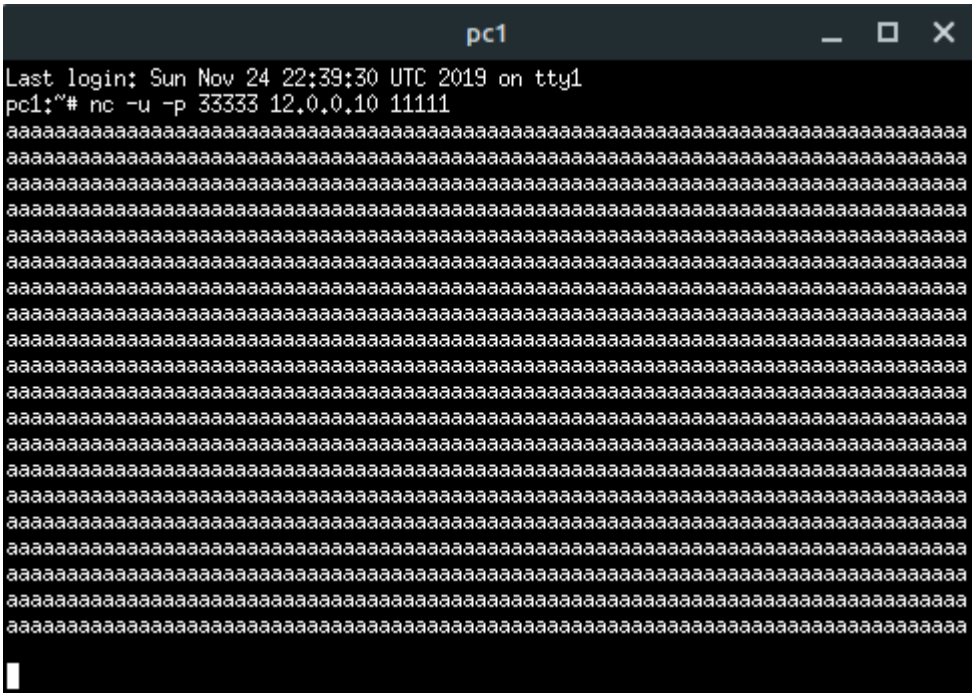


Figura 2: trafico UDP

Antes de observar en la captura lo que ha ocurrido responde a estas preguntas:

1. cuántos datagramas UDP crees que se han enviado, y por qué?
   Se enviara 1 datagrama UDP porque cada vez que una aplicación envia datos produce un unico datagrama.
2. cuántos datagramas IP crees que se han enviado, y por qué?
   Se enviaran 2 datagramas debido a que el total de bytes a enviar excede los 1480 de la seccion de datos del datagrama ip en ethernet.
3. cuántos bytes de datos irán en cada datagrama UDP?
   1480 en el primero y 120 en el segundo.

Interrumpe ahora la captura y comprueba si tus suposiciones son correctas. Explica razonadamente el número de datagramas UDP, el número de datagramas IP, y el número de bytes de datos que va en cada uno de los datagramas UDP.

1.2.3. UDP es un protocolo basado en datagramas: no hay cierre de conexión

1. Inicia una captura en el router r1 . Esta vez no es necesario guardar el contenido en un fichero.

2. Interrumpe la ejecución del cliente pulsando la tecla C mientras mantienes pulsada la tecla Ctrl (a partir de ahora diremos pulsa Ctrl+C).

3. Interrumpe la ejecución del servidor pulsando Ctrl+C .
Explica cuántos paquetes deberían haberse capturado y por qué como consecuencia de terminar el cliente con Ctrl+C . Interrumpe la captura y comprueba tu suposición.

Se capturaron 0 paquetes debido a que UDP no realiza cierre de conexión no es necesario enviar nada.

1.2.4. Buffer de recepción en UDP:

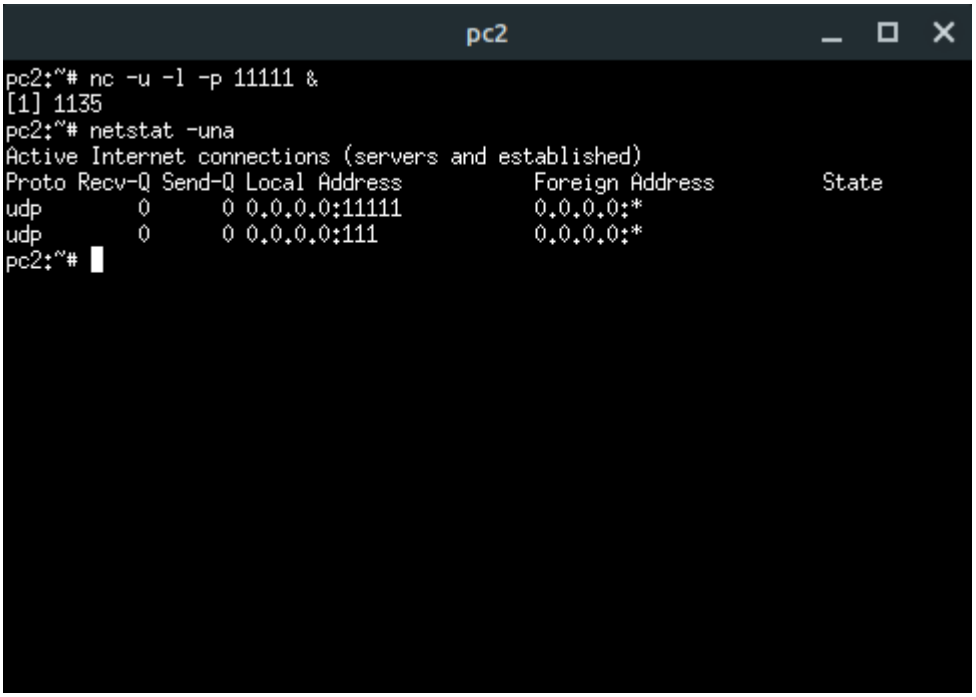
Los protocolos de nivel de transporte trabajan tanto en TCP como en UDP con buffers in/out (recepción/envío) asociados a cada puerto (en el caso de TCP asociados a cada conexión). Cuando llega un datagrama UDP o un segmento TCP, se almacenan los bytes para la aplicación en el buffer in correspondiente. Cuando una aplicación envía bytes a través de un puerto UDP o de una conexión TCP, se almacenan sus bytes en el buffer out asociado a dicho puerto/conexión.

Para ver estos buffers vamos a utilizar la herramienta netstat .

1. Lanza en segundo plano una aplicación servidor utilizando nc en la máquina 12.0.0.10 que reciba mensajes UDP destinados al puerto 11111 en segundo plano:

```
nc -u -l -p 11111 &
```

2. Observa el estado que muestra netstat -una en el servidor para la aplicación que acabas de arrancar y sus buffers.

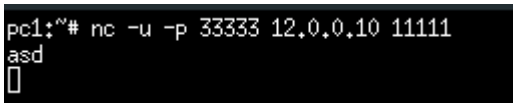


3. Trae a primer plano la ejecución de la aplicación servidor arrancada con nc , ejecutando para ello en el terminal del servidor esta orden: fg

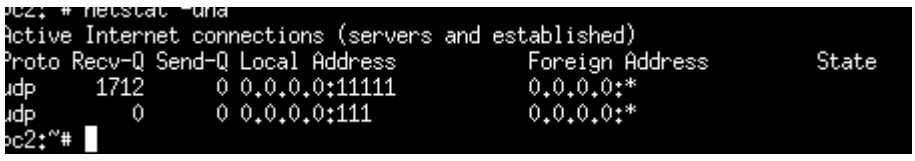
Tras traer a primer plano una aplicación, ésta recupera la entrada estándar del teclado. Ahora por tanto lo que se teclee se le envía a la aplicación servidor.

4. Pausa con Ctrl+Z la ejecución de la aplicación servidor nc . De esta forma la aplicación en el servidor siga arrancada pero no esté ejecutándose (la CPU no ejecutará instrucciones de la aplicación servidor mientras esté suspendida su ejecución). Mientras está suspendido el servidor, si la implementación de UDP en la máquina del servidor recibe datos destinados a la aplicación servidor, estos datos se almacenarán en el correspondiente buffer in , y no van a ser leídos por la aplicación nc porque su ejecución está suspendida temporalmente.

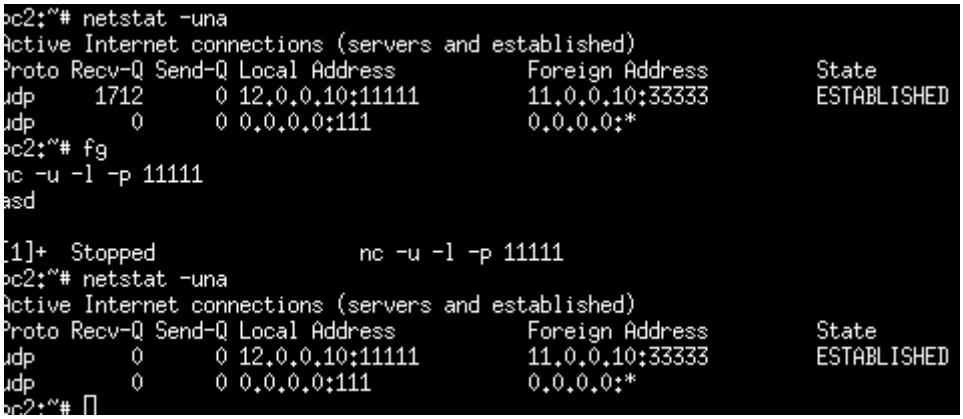
5. Para ver cómo los datos se quedan almacenados en el buffer in de UDP del puerto en el que escucha el servidor, envía unos cuantos caracteres desde el cliente y pulsa la tecla INTRO o ENTER .



6. Ejecuta netstat -una en el servidor para ver cómo esos datos se quedan en el buffer de recepción y no los lee la aplicación. Verás que la cantidad muestra 1712 bytes y no la cantidad de caracteres que has introducido desde el cliente. Esto es debido a que en UDP se reserva este cantidad en el buffer por cada paquete recibido.



7. Ahora vamos a volver a activar el servidor, que estaba suspendido con Ctrl+Z , trayéndole a primer plano. Ejecuta para ello la orden fg . Verás como inmediatamente los datos que habías enviado desde el cliente se muestran en la pantalla: la aplicación servidor los ha leído del buffer de recepción, que se ha vaciado, y los ha mostrado en la pantalla.



8. Una vez realizada la prueba puedes interrumpir la ejecución del cliente y el servidor con Ctrl+C .

1.2.5. Errores en las comunicaciones UDP

Provoca las siguientes situaciones de error:

1. Existe la máquina 12.0.0.10, pero en ella no hay una aplicación servidor escuchando en el puerto 11111. Inicia una captura en el router r1 . Prueba a lanzar el cliente y envía datos desde el cliente. Interrumpe la captura.
Explica razonadamente los paquetes capturados.

- Lo primero que ocurre es que el pc1 desea averiguar quien tiene la ip 11.0.0.1 para poder enviar el paquete atravez de su puerta de enlace predeterminada.

- Posteriormente recibe la respuesta proveniente del router que le dice que el tiene esa ip.

- Pc1 procede a enviar el paquete UDP con el mensaje al destino 12.0.0.10 , se lo enviá al router para que este se lo envié al destino.

- La PC2 con ip 12.0.0.10 le devuelve un mensaje de ICMP con tipo 3(destino inalcanzable) y código 3 (Puerto inalcanzable) dado que el puerto al que el mensaje estaba destinado no esta disponible para recibirlo debido a que ninguna aplicación esta escuchándolo.

2. Existe la red 12.0.0.0/24 y hay ruta para llegar hasta ella, pero no existe la máquina 12.0.0.10 (para realizar este apartado apaga la máquina 12.0.0.10). Inicia una captura en el router r1 . Prueba a lanzar el cliente y envía datos desde el cliente. Interrumpe la captura y comprueba su contenido.

El primer paquete es el envió del mensaje UDP con destino 12.0.0.10 el cual se envia a la eth0 del router

El segundo paquete capturado es de tipo ICPM de tipo destino inalcanzable (3) y con el código 1 host inalcanzable. Lo cual ocurre porque no hay un equipo con la ip 12.0.0.10 en la red que reciba el paquete.

2. Comunicación de aplicaciones usando el protocolo TCP

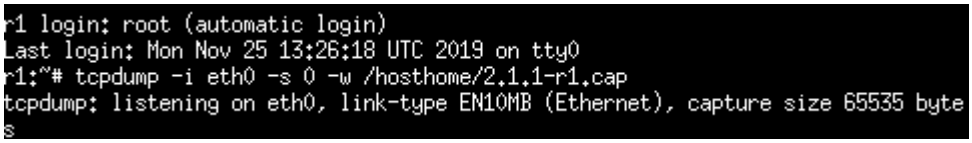
En este apartado utilizarás la orden nc para observar el funcionamiento de TCP en diversas situaciones. Ve ahora al anexo de esta práctica en el que se explica cómo utilizar nc para arrancar clientes y servidores TCP, y vuelve aquí después para continuar.

Utilizaremos el mismo escenario NetGUI del apartado anterior.

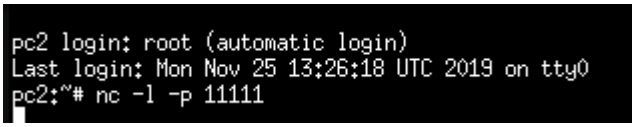
2.1. TCP es un protocolo orientado a conexión.

2.1.1. Establecimiento de conexión

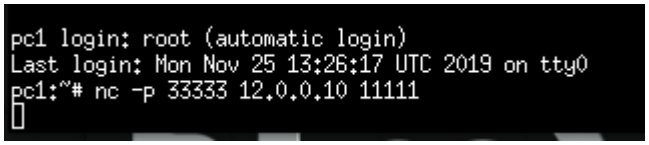
1. Inicia una captura en el router r1 y guarda su contenido en un fichero.



2. Usando nc, lanza una aplicación servidor en la máquina 12.0.0.10 que atienda peticiones de conexión destinadas al puerto TCP 11111: nc -l -p 11111



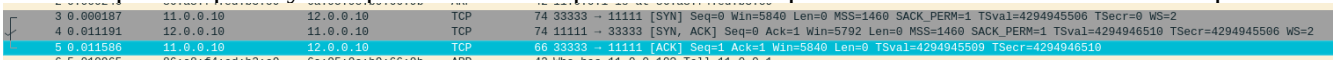
3. Lanza una aplicación cliente con nc en la máquina 11.0.0.10 para que se establezca una conexión TCP con la aplicación servidor, usando como puerto local TCP el 33333: nc -p 33333 12.0.0.10 11111 . Explica cuántos paquetes deberían haberse capturado y por qué.



3 paquetes debido a que son los mínimos necesarios para el proceso de conexión inicial.

- el primero seria el sync enviado por el cliente (pc1)
- El segundo es sync ack en respuesta al cliente desde el servidor (pc2).
- El ACK con la información del numero de secuencia y ventana que el servidor le propone al cliente.

Interrumpe la captura y comprueba si tus respuestas se corresponden con lo observado en la captura.



2.1.2. Cierre de conexión

1. Inicia una captura en el router r1 y guarda el contenido en un fichero.

2. Interrumpe la ejecución del cliente pulsando Ctrl+C en el cliente 3 . Explica cuántos paquetes deberían haberse capturado y por qué.
Deberían de haberse capturado 3 paquetes.

- La petición de cierre de la conexión (Fin, ACK)
- La confirmación de la recepcion del cierre de conexión
- La confirmación de los paquetes recibidos por el servidor que aun no habían sido informados como recepcionados.

Interrumpe la captura y comprueba si tus respuestas se corresponden con lo observado en la captura.

2.2. Buffer de recepción en TCP

Vamos a visualizar los bufferes in/out (recepción/emisión) en TCP. Inicia una captura en el terminal de r1 guardando el contenido en un fichero.

1. Lanza una aplicación servidor utilizando nc en la máquina 12.0.0.10 que acepte conexiones en el puerto TCP 11111 (arráncala en segundo plano): nc -l -p 11111 &

```
pc2 login: root (automatic login)
Last login: Tue Nov 26 00:40:57 UTC 2019 on tty0
pc2:~# nc -l -p 11111 &
[1] 722
pc2:~#
```

2. Lanza una aplicación cliente con nc en la máquina 11.0.0.10 para que se conecte al servidor, desde el puerto local TCP 33333: nc -p 33333 12.0.0.10 11111

```
pc1 login: root (automatic login)
Last login: Mon Nov 25 14:10:19 UTC 2019 on tty1
pc1:~# nc -u -p 33333 12.0.0.10 11111
```

3. Observa el estado que muestra netstat -tna en el servidor y sus buffers.

```
pc2:~# netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:11111           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
```

4. Trae a primer plano la ejecución de nc ejecutando en el terminal: fg

5. Pausa con Ctrl+Z la ejecución de nc en el servidor, para que la aplicación del servidor siga arrancada pero no se ejecute, por tanto, si TCP en el lado servidor recibe datos, estos no se se van a leer en nc quedarán almacenados en la cola de entrada de la implementación de TCP.

```
nc -l -p 11111

[1]+  Stopped                 nc -l -p 11111
pc2:~#
```

6. Para ver cómo los datos se quedan almacenados en el servidor, envía una cadena de caracteres desde el cliente y pulsa INTRO .

```
pc1:~# nc -p 33333 12.0.0.10 11111
hola
```

7. Ejecuta netstat -tna en el servidor para ver cómo esos datos se quedan en el buffer de recepción y no los lee la aplicación.

```
c2:~# netstat -tna
ctive Internet connections (servers and established)
roto Recv-Q Send-Q Local Address           Foreign Address         State
cp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
cp        5      0 12.0.0.10:11111         11.0.0.10:33333         ESTABLISHED
c2:~#
```

8. Interrumpe la captura y fíjate cómo hay un asentimiento que indica que todos los datos han sido recibidos. Dado que la aplicación servidor está suspendida, los datos se encuentran almacenados en el buffer de recepción de la implementación de TCP, en el kernel del sistema operativo, pero no los ha leído aún la aplicación servidora arrancada con nc.

```
11 17.297104    12.0.0.10          11.0.0.10          TCP        66 11111 → 33333 [ACK] Seq=1 Ack=6 Win=5792 Len=0 TSval=4294944075 TSecr=1326
```

9. Trae a primer plano la ejecución del servidor, para ello usa fg . Verás como los datos que habías enviado desde el cliente se muestran en la pantalla. El servidor los ha leído del buffer de recepción y el buffer está vacío.

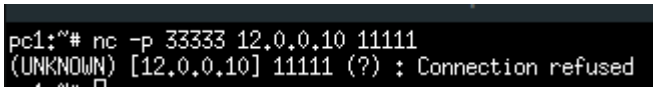
```
c2:~# fg
c -l -p 11111
ola

1]+  Stopped                 nc -l -p 11111
c2:~# netstat -tna
ctive Internet connections (servers and established)
roto Recv-Q Send-Q Local Address           Foreign Address         State
cp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
cp        0      0 12.0.0.10:11111         11.0.0.10:33333         ESTABLISHED
c2:~#
```

Una vez realizada la prueba puedes interrumpir la ejecución del cliente y el servidor.

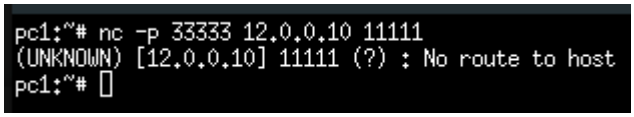
2.3. Errores en las comunicaciones TCP

Provoca las siguientes situaciones de error:

1. Existe la máquina 12.0.0.10 pero no hay una aplicación escuchando en el puerto 11111. Prueba a lanzar el cliente y comprueba qué ocurre.



2. Existe la red 12.0.0.0 y hay ruta para llegar hasta ella, pero no existe la máquina 12.0.0.10. (Para realizar este apartado apaga la máquina 12.0.0.10). Prueba a lanzar el cliente y comprueba qué ocurre.



2.4. Análisis inicial de la captura de TCP

En la captura tcp.cap se muestra una comunicación TCP entre dos aplicaciones.

Contesta a las siguientes preguntas:

1. Cuál es la dirección IP y el puerto del cliente TCP y la dirección IP y el puerto del servidor TCP?

	Cliente	Servidor
IP	10.0.0.2	11.0.0.2
Puerto	60709	34000

2. Cuántos segmentos TCP se han enviado desde el cliente al servidor? 6

3. Cuántos segmentos TCP se han enviado desde el servidor al cliente? 4

4. Indica qué extremo cierra antes la conexión. El Cliente.

2.5. Números de secuencia

Sigue analizando la captura tcp.cap .

En el menú de Wireshark , seleccionando en el menú Edit → Preferences → Protocols → TCP , puedes desactivar la opción Relative Sequence Numbers & Window Scaling . De esta forma podrás observar los números de secuencia reales, en lugar de los números relativos que muestra por omisión Wireshark .

1. Cuántos bytes de datos envía el servidor al cliente? Razona la respuesta. Indica cuáles son los números de secuencia del SYN y del FIN que envía el servidor, y qué relación tienen con la cantidad de datos enviada por el servidor al cliente.

2. Cuántos bytes de datos envía el cliente al servidor? Razona la respuesta. Indica cuáles son los números de secuencia del SYN y del FIN que envía el cliente, y qué relación tienen con la cantidad de datos enviada por el cliente al servidor.

Cuando hayas observado los números de secuencia reales, vuelve a activar la opción Relative Sequence Numbers & Window Scaling para que resulte más fácil analizar la captura en los siguientes apartados.

2.6. RTT

Sigue analizando la captura tcp.cap .

1. Para cada uno de los segmentos de datos que envía el cliente al servidor, indica cuál es el RTT. Observa para ello los tiempos de envío de los segmentos y los de recepción de sus correspondientes asentimientos.

# de paquete TCP Cliente que representa el segmento.	RTT en milisegundos
3	9
6	1
8	1
10	2

2.7. Factor de escala sobre la ventana anunciada

Sigue analizando la captura tcp.cap . En los segmentos que llevan activado el flag SYN se informa de los valores de número de secuencia inicial y ventana inicial anunciada que tiene cada extremo de la comunicación TCP. La opción factor de escala de la ventana (window scale) puede indicarse en la parte de opciones de los segmentos que llevan el flag SYN.

Cuando el lado que abre la conexión quiere utilizar un factor de escala para la ventana anunciada, activará la opción en su paquete SYN. El otro lado debe activar la opción en su paquete SYN+ACK para indicar que entiende esta opción de TCP y va a utilizarla también. A partir de este momento, en los segmentos que envíen ambos lados se aplicará el factor de escala para calcular el valor real de la ventana que se está anunciando: se multiplicará 2 factor por el campo de ventana anunciada que viaja en el segmento. Nótese que sólo en el segmento SYN y en el SYN+ACK viaja el factor de escala, pero éste no se aplica al campo ventana anunciada de estos dos segmentos, sino que se aplica a todos los demás segmentos enviados.

Wireshark aplica automáticamente el factor de escala y lo indica con la etiqueta scaled . Para ver el campo ventana anunciada que viaja realmente en la cabecera TCP hay que desactivar la opción Relative Sequence Numbers & Window Scaling en el menú Edit → Preferences → Protocols → TCP .

1. Indica cuál es el valor del campo de ventana anunciada (Window size) en los segmentos SYN en cada uno de los dos sentidos.

El equipo cliente ip 10.0.0.2 tiene un tamaño de ventana de 5840 y el servidor con ip 11.0.0.2 5792

2. Indica cuál es el valor del campo de ventana anunciada (Window size) en los segmentos que no tienen el flag SYN activado en cada uno de los dos sentidos

# de paquete que representa al segmento desde el Cliente	Window Size
5	2920
6	2920
8	2920
10	2920
12	2920

3. Indica cuál es el valor real de ventana anunciada teniendo en cuenta el factor de escala que se incluye en las opciones de los segmentos SYN y SYN+ACK.

5840

Cuando hayas observado el campo de la ventana anunciada, vuelve a activar la opción Relative Sequence Numbers & Window Scaling .

2.8. MSS

En la captura tcp-mss-pmtu.cap se muestra el principio de una comunicación TCP. Ordena en Wireshark los paquetes por la columna Time . Contesta a las siguientes preguntas:

1. Indica cuál es el valor anunciado de MSS en las cabeceras opcionales de los paquetes SYN de los dos sentidos de la conexión. Dados estos dos valores, indica qué tamaño de datos crees usarán ambos sentidos de la conexión si tienen que enviar datos.

- El equipo con la ip 11.0.0.11 tiene un MSS de 660
- El equipo con la ip 12.0.0.12 anuncia un mss de 560

Ambos equipos usaran un valor igual o menor a 560.

2. Mira los tamaños de las cabeceras de los distintos segmentos, medidos en palabras de 4 bytes y razona por qué no todos los segmentos tienen el mismo tamaño. Crees que el tamaño de la cabecera puede influir en el tamaño máximo de datos que posteriormente utilizará TCP para enviar segmentos?

Si puede influir.

3. Teniendo en cuenta que en el instante en el que se envía el segmento número 4, la máquina 11.0.0.11 tenía más de 2.000 bytes de datos que enviar a la máquina 12.0.0.12, por qué envía menos? Cuántos bytes envía en ese segmento 4? Cómo se justifica el número de bytes de datos que contiene el segmento 4 dados los valores de MSS anunciados en los segmentos SYN?

Enviá menos debido a que el MSS anunciado por la maquina 12.0.0.12 es de 560 no puede enviar mas que esa cantidad. En ese segmento enviá 548bytes. Se justifica debido a que que se están restando de los 560 bytes máximos los 12 bytes de opciones agregados a la cabecera TCP.

4. Explica qué tipo de paquete son los paquetes 5 y 7 de la captura. Qué significan y cuál puede ser la razón de que se hayan enviado? Qué campo de la cabecera IP de los paquetes 4 y 6 ha provocado que se envíen los paquetes 5 y 7?

Los paquetes 5 y 7 son del tipo de ICMP, y significa que el paquete al que hacen referencia no fueron entregados a su destinatario. Debido a que el siguiente dispositivo en la red no acepta paquetes mayores a 500 bytes,

5. Mira el paquete 8 de la captura. Por qué se retransmite este segmento? Comprueba el tamaño de su campo de datos. Explica la relación de este valor con la información contenida en los paquetes 5 y 7.
Se retransmite dado que se informo que el paquete no fue entregado y tcp es un protocolo fiable. Tiene un tamaño de datos de 500 bytes. Y se debe a que anteriormente se le especifico que un elemento en la red tiene establecido como cantidad máxima de datos. Ese valor.

2.10. Funcionamiento básico de la ventana anunciada

En la captura tcp-window.cap se muestra el principio de una comunicación TCP. Para este apartado, utiliza en Wireshark los números de secuencia relativos al principio de la conexión.

Ayudándote de la gráfica tcptrace , contesta a las siguientes preguntas relativas al sentido de comunicación 12.0.0.100 → 13.0.0.100:

1. Cuál es el número de secuencia del primer byte de datos contenido en el paquete número 6?
1

2. Cuál es el número de secuencia del último byte de datos contenido en el paquete número 6?
548

3. El paquete número 7, cuántos bytes de datos asiente? 548

4. En el momento de enviar el paquete número 8, cuál es el último valor de ventana anunciada por el receptor que ha recibido el emisor? En qué número de paquete venía anunciado? Cuántos bytes de datos puede enviar como máximo el emisor en ese momento, en uno o más segmentos, antes de volver a recibir otro ACK del receptor?

El valor de ventana anunciado es de 2592 y venia anunciado en el paquete numero 3. en ese momento puede enviar un máximo de 1295 bytes antes de llenar la venta.

5. En el momento de enviar el paquete número 13, cuál es el último valor de ventana anunciada por el receptor que ha recibido el emisor? En qué número de paquete venía anunciado? Cuántos bytes de datos puede enviar como máximo el emisor en ese momento, en uno o más segmentos, antes de volver a recibir otro ACK del receptor?

El ultimo valor anunciado por el receptor es de 2192, que fue anunciado en el paquete numero 9, y por lo tanto puede enviar un maximo de 350 bytes.

6. Podría haber enviado el emisor un segmento con datos nuevos en el instante 0.321000 segundos? Por qué?

Si podría enviar datos debido a que aun puede enviar 350 bytes antes de llegar al valor maximo de la ventana.

7. Identifica en la gráfica tcptrace de este sentido de la comunicación en qué otro periodo de tiempo se produce una situación similar a la que ocurre al enviarse el paquete número 13.
Se produce en el periodo 0.423846 seg.

2.11. Retransmisiones y asentimientos

En la captura tcp-timeout-probes.cap se muestra el principio de una comunicación TCP. Para este apartado, utiliza en Wireshark los números de secuencia relativos al principio de la conexión.

Ayudándote de la gráfica tcptrace , contesta a las siguientes preguntas relativas al sentido de comunicación 12.0.0.100 → 13.0.0.100:

1. El paquete número 16 es una retransmisión, aunque Wireshark no lo etiqueta como tal:

a) Mirando la gráfica tcptrace , cómo puede identificarse que dicho paquete 16 es una retransmisión? Debido a la secuencia del paquete.
b) Qué número de paquete es la primera transmisión de los bytes de datos que viajan en este paquete 16? paquete numero 5
c) Cuál ha sido el plazo de retransmisión aplicado a esa primera transmisión del paquete? Ha sido de 0.208722 seg.
d) Mirando la gráfica tcptrace ,vuelve a ser retransmitido este paquete en la conexión? Con qué plazo de retransmisión? Por qué? Si vuelve a ser retransmitido con un plazo de 0.444351 seg., debido a que cuando la transmisión falla se dobla el plazo para la siguiente retransmisión.

2. Identifica en la gráfica tcptrace otros segmentos que son retransmisión.

El paquete 45 es uno de los que es una retransmisión.

3. En el momento de transmitir el paquete 18, cuántos bytes están transmitidos y pendientes de que llegue su asentimiento? Y cuántos paquetes (indica su número de paquete)? 7570 bytes están transmitidos y pendientes de asentar. Y 5 paquetes (6,9,10,1115 y 16)

4. El paquete 19 es un asentimiento. Cuántos bytes asiente? Y cuántos segmentos? Asienta 6056 bytes en 4 segmentos.

5. Identifica con ayuda de la gráfica tcptrace otros asentimientos que asienten varios paquetes a la vez. Es el caso del asentamiento del paquete numero 46. que asienta 3 segmentos.

2.12. Sondas de ventana

Sigue analizando la captura tcp-timeout-probes.cap . Responde a las siguientes cuestiones.

1. Identifica en la gráfica tcptrace un periodo de tiempo en que el servidor está anunciando ventana 0 al cliente. Identifica en la lista de paquetes los que son enviados y recibidos en ese periodo de tiempo. Por qué el cliente no envía bytes de datos en ese periodo de tiempo? Porque si se le ha anunciado una

ventana de 0 no tiene sentido enviar datos debido a que el buffer del otro equipo esta lleno, enviarle datos solo ocasionaría perdidas ya que no hay forma de almacenarlos en el equipo que los recibira.

2. Qué números de paquete son las sondas de ventana? Cómo los identifica wireshark ? 63,65 y 67 los identifica como TCP ZeroWindow

3. Cuánto tiempo pasa desde que el cliente recibe el primer anuncio de ventana 0 hasta que envía la primera sonda de ventana? 0.484753 seg.

4. Cuánto tiempo pasa desde que el cliente recibe el segundo anuncio de ventana 0 hasta que envía la segunda sonda de ventana? 0.878449 seg.

5. Qué paquete es el que anuncia al cliente que la ventana vuelve a estar abierta? Cuántos bytes de datos puede enviar el cliente a partir de ese momento? Entre qué números de secuencia? El paquete numero 68, y el cliente puede enviar 5792 bytes de datos entre el numero de secuencia 1 y 5792