Práctica 4: UDP, TCP

Arquitectura de Redes de Ordenadores
Arquitectura de Internet
versión 2

GSyC
Departamento de Teoría de la Señal y Comunicaciones y Sistemas Telemáticos y Computación

Abril de 2017

Resumen

En esta práctica se aprende el funcionamiento básico de los protocolos de nivel de transporte UDP y TCP.

Nota: Al cargar capturas en *Wireshark* es necesario ordenar los paquetes por su marca de tiempo, pulsando en la pestaña `Time`, de esta forma podremos analizar lo que ha ocurrido ordenadamente siguiendo el eje temporal.

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?
2. ¿Qué información puedes extraer de la captura sobre la red en la que se ha realizado la captura?
3. ¿Cuál es el número de paquetes UDP y número de bytes de datos intercambiados?

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

Descarga de la página de la asignatura el fichero `lab-p4.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-p4` en la que está el escenario. Verás aparecer la red de la figura 1.

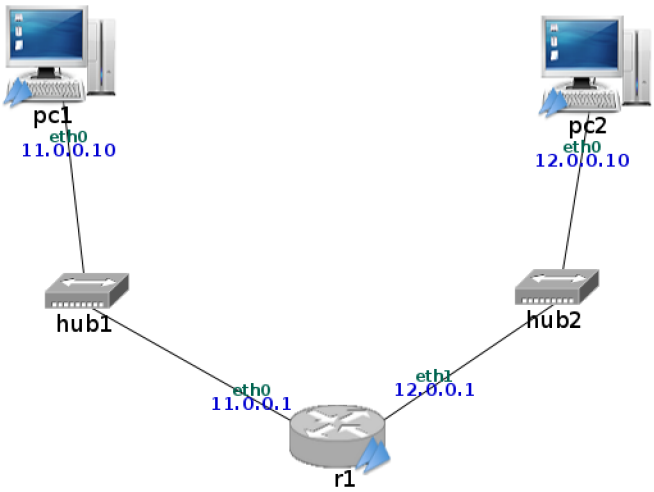

Figura 1: Escenario lab-p4

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].
2. Usando `nc` lanza una aplicación servidor UDP en la máquina 12.0.0.10 y puerto 11111: `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`
4. Interrumpe la captura.

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

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.
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 figura 2).

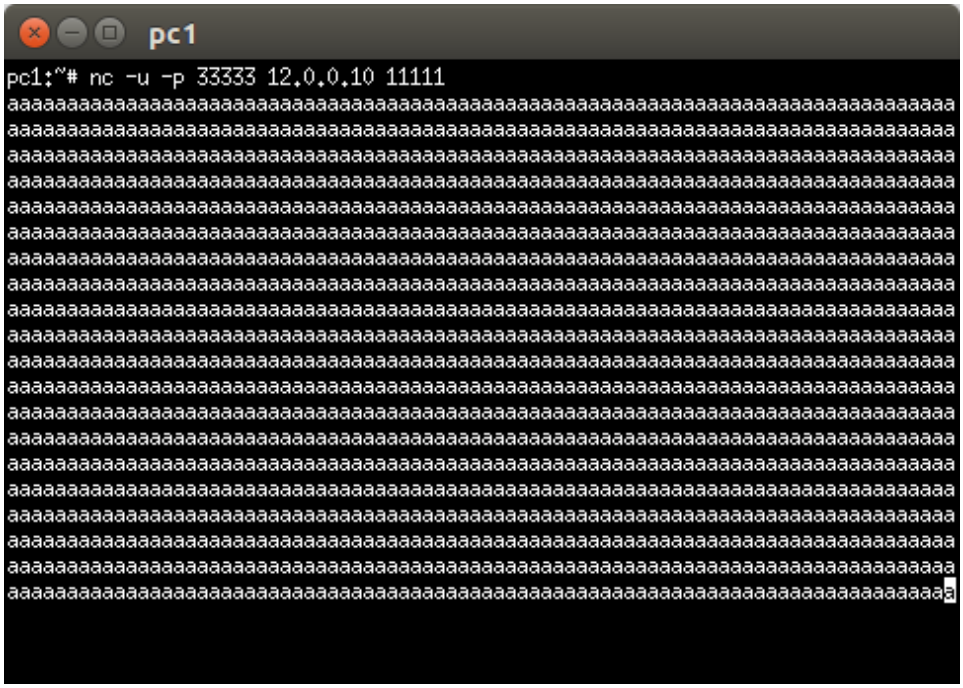

Figura 2: Tráfico UDP

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?
2. ¿cuántos datagramas IP crees que se han enviado, y por qué?
3. ¿cuántos bytes de datos irán en cada datagrama UDP?

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]Esta captura puedes realizarla sin necesidad de guardar el contenido en un fichero. La utilizaremos para ver los paquetes que se generan. Para ello ejecuta en r1: tcpdump -i eth0.

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.

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[2] 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`, trayéndolo 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.
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.

[2] Añadiendo un carácter & al final de una orden introducida en la shell se ejecuta dicha orden en segundo plano, lo que significa que la entrada del teclado no quedará ligada al proceso arrancado, pudiéndose así seguir ejecutando comandos en la shell a la vez que se está ejecutando concurrentemente la orden ejecutada en segundo plano

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.

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.

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 &`
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`
3. Observa el estado que muestra `netstat -tna` en el servidor y sus buffers.
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.
6. Para ver cómo los datos se quedan almacenados en el servidor, envía una cadena de caracteres desde el cliente y pulsa `INTRO`.
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.
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.
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.

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

[3] Con nc, una vez que se interrumple la conexión desde el cliente, el servidor también cierra la conexión.
Con otras aplicaciones podría mantenerse abierta la conexión desde el servidor si éste tuviera más datos que enviar

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?
2. ¿Cuántos segmentos TCP se han enviado desde el cliente al servidor?
3. ¿Cuántos segmentos TCP se han enviado desde el servidor al cliente?
4. Indica qué extremo cierra antes la conexión.

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.

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.

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 [4]

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.

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.

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?

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?

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?

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.

2.9. Retransmisión de SYN

En la captura `tcp-syn.cap` se muestra una comunicación TCP. Contesta a las siguientes preguntas:

1. ¿Cuántas veces se envía el segmento SYN del cliente al servidor?

2. Indica la secuencia de marcas de tiempo de los segmentos SYN enviados, y explica sus valores atendiendo al mecanismo *exponential backoff*.

3. Explica por qué aparecen varios segmentos ACK repetidos.

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?

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

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

[4] TCP inicialmente no anuncia en el campo Window size todo el buffer que tiene disponible, sigue un algoritmo para anunciar inicialmente un valor pequeño y lo va aumentando según va avanzando la conexión hasta que llega a anunciar el tamaño total del buffer.

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?

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?

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

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.

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?

   b) ¿Qué número de paquete es la primera transmisión de los bytes de datos que viajan en este paquete 16?

   c) ¿Cuál ha sido el plazo de retransmisión aplicado a esa primera transmisión del paquete?

   d) Mirando la gráfica `tcptrace`, ¿vuelve a ser retransmitido este paquete en la conexión? ¿Con qué plazo de retransmisión? ¿Por qué?

2. Identifica en la gráfica `tcptrace` otros segmentos que son 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)?

4. El paquete 19 es un asentimiento. ¿Cuántos bytes asiente? ¿Y cuántos segmentos?

5. Identifica con ayuda de la gráfica `tcptrace` otros asentimientos que asienten varios paquetes a la vez.

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?

2. ¿Qué números de paquete son las sondas de ventana? ¿Cómo los identifica *wireshark*?

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

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

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?

3. Anexo: Funcionamiento de nc

Las aplicaciones que se utilizarán en esta práctica para generar tráfico TCP y UDP siguen el modelo de comunicaciones cliente/servidor. En este modelo, cuando dos aplicaciones se comunican una de ellas funcionará como servidor y la otra funcionará como cliente.

Siempre es necesario lanzar primero la aplicación que funciona como servidor, que quedará a la espera de recibir tráfico procedente de la aplicación cliente, que se deberá lanzar después.

En esta práctica utilizaremos la aplicación `nc` para arrancar aplicaciones que utilizan TCP o UDP. `nc` puede arrancarse como cliente o como servidor, utilizando TCP o UDP como protocolo de transporte. Una aplicación `nc` lanzada como cliente se comunicará con otra lanzada como servidor y viceversa. Una aplicación lanzada con `nc` como cliente lee de la entrada estándar (por omisión el teclado) los caracteres introducidos y al pulsar la tecla `INTRO` la línea de texto es enviada usando TCP o UDP a la aplicación servidor. Al recibir la línea de texto, la aplicación servidor lanzada con `nc` mostrará en la pantalla los datos recibidos de la aplicación cliente lanzada con `nc`.

3.1. Tráfico UDP

3.1.1. Aplicación servidor UDP

Para arrancar una aplicación que funciona como servidor utilizando el protocolo UDP ejecutaremos la siguiente orden:

```
nc -u -l -p <Pto-Loc>
```

Donde:

- `<Pto-Loc>` es el número de puerto local UDP en el que la aplicación servidor esperará recibir los datagramas UDP de una aplicación cliente.

Por ejemplo, si queremos arrancar una aplicación servidor UDP en el puerto 7777 de la máquina `pc1` utilizaremos la siguiente orden:

```
pc1:~# nc -u -l -p 7777
```

3.1.2. Aplicación cliente UDP

Para arrancar una aplicación que funciona como cliente utilizando el protocolo UDP ejecutaremos la siguiente orden:

```
nc -u -p <Pto-Loc> <IP-dest> <Pto-dest>
```

Donde:

- `<Pto-Loc>` es el número de puerto local UDP en el que la aplicación cliente esperará recibir los datagramas UDP que vengan del servidor.
- `<IP-dest>` es la dirección IP de la máquina donde se está ejecutando la aplicación servidor UDP.
- `<Pto-dest>` es el número de puerto UDP en el que escucha la aplicación servidor UDP.

Por ejemplo, si queremos arrancar una aplicación cliente UDP que espere recibir datagramas UDP en el puerto 6666 y que envíe datagramas UDP a la dirección IP 200.0.0.1 y puerto 7777 (donde se encuentra esperando recibir datagramas UDP la aplicación servidor) utilizaremos la siguiente orden:

```
pc2:~# nc -u -p 6666 200.0.0.1 7777
```

3.1.3. Envío de datos UDP

Una vez lanzadas las aplicaciones servidor UDP y cliente UDP, el cliente puede enviarle líneas de texto al servidor. Después de que el cliente haya enviado al menos una línea de texto al servidor, todo lo que escribamos a través de la entrada estándar de un extremo será enviado al otro extremo como datagramas UDP: si escribimos en el terminal de la aplicación cliente, esto será enviado a la aplicación servidor, y viceversa.

Para interrumpir la ejecución de estas aplicaciones se debe utilizar `Ctrl+C`.

3.2. Tráfico TCP

3.2.1. Aplicación servidor TCP

Para arrancar una aplicación que funciona como servidor utilizando el protocolo TCP ejecutaremos la siguiente orden:

```
nc -l -p <Pto-Loc>
```

Donde:

- `<Pto-Loc>` es el número de puerto local TCP en el que la aplicación servidor esperará recibir mensajes TCP de una aplicación cliente.

Por ejemplo, si queremos arrancar una aplicación servidor TCP en el puerto 7777 de la máquina `pc1` utilizaremos la siguiente orden:

```
pc1:~# nc -l -p 7777
```

3.2.2. Aplicación cliente TCP

Para arrancar una aplicación que funciona como cliente utilizando el protocolo TCP ejecutaremos la siguiente orden:

```
nc -p <Pto-Loc> <IP-dest> <Pto-dest>
```

Donde:

- `<Pto-Loc>` es el número de puerto local TCP en el que la aplicación cliente esperará recibir los mensajes de la aplicación servidor TCP.
- `<IP-dest>` es la dirección IP de la máquina donde se está ejecutando la aplicación servidor TCP.
- `<Pto-dest>` es el número de puerto TCP en el que escucha la aplicación servidor TCP.

Por ejemplo, si queremos arrancar una aplicación cliente TCP que utilice el puerto origen 6666 para establecer una conexión TCP con un servidor TCP que escuche en el puerto destino 7777 de la máquina 200.0.0.1, utilizaremos la siguiente orden:

```
pc2:~# nc -p 6666 200.0.0.1 7777
```

3.2.3. Envío de datos TCP

Una vez iniciada la aplicación servidor TCP, ésta se queda esperando recibir mensajes de una aplicación cliente TCP.

Una vez iniciada la aplicación cliente TCP, ésta intercambiará unos mensajes de control (apertura de conexión) con la aplicación servidor, por lo que es imprescindible que dicha aplicación servidor haya sido lanzada antes.

Si la comunicación entre ambas aplicaciones es posible, a partir de este momento todo lo que escribamos a través de la entrada estándar de una aplicación será enviada a la otra: si escribimos en el terminal de la aplicación cliente, esto será enviado a la aplicación servidor, y viceversa.

Para interrumpir la ejecución de estas aplicaciones se debe utilizar `Ctrl+C`.