Práctica 3 - Midiendo retardos en Ethernet

1. Introducción

El tiempo que tarde la información que quiera enviar una aplicación a otra atravesando una red de conmutación de paquetes depende del tiempo que le cueste a cada paquete atravesar la red y de cómo se secuencien entre ellos los paquetes enviados. En esta práctica nos vamos a centrar en el tiempo que tarde un paquete en atravesar un camino por una red. Este tiempo va a depender del tiempo que se tarde en transmitir por cada enlace, del que tarde en propagarse la señal por cada medio y del que pase el paquete en cada uno de los equipos de conmutación (mientras es procesado o mientras espera a que se le asigne el medio de salida y pueda comenzar su transmisión). En general, en redes de área local como la que vamos a emplear, los tiempos de propagación van a ser muy pequeños y difíciles de medir (por debajo del microsegundo). Los tiempos de transmisión van a depender del tamaño de los paquetes y de la tasa de transmisión.

Dado que vamos a emplear los PCs del laboratorio para medir, debemos entender que el propio proceso de medida va a introducir errores. Por ejemplo, si se está enviando una trama Ethernet de 64 bytes por un enlace a 100Mb/s el tiempo de transmisión de la misma será de 5.12μs. Necesitaríamos que la marca de tiempo que indica cuándo se recibe un paquete fuera puesta por la tarjeta de red, ya que si esperamos a que el paquete sea copiado a la memoria del sistema operativo es probable que cualquier otra tarea que esté llevando a cabo la CPU retrase la atención a la tarjeta de red e introduzca un error en esa marca de tiempo, posiblemente de órdenes de magnitud superiores. Si queremos medir tiempos en diferentes máquinas nos encontraremos también con que los relojes de estas no están sincronizados con la precisión necesaria para medidas tan finas. Por no complicar el proceso de medida intentaremos llevar a cabo la práctica con mecanismos simples, entendiendo siempre que algunas medidas pueden tener error en los tiempos por estos y otros fenómenos.

El objetivo de esta práctica es familiarizarse con las dependencias de los diferentes retardos con parámetros como los tamaños de los paquetes, las tasas de transmisión o el número de saltos.

Emplearemos uno solo de los PC (PC A, B o C). Enviaremos tramas Ethernet por uno de sus interfaces para que sigan un camino por una LAN hasta llegar a otro de los interfaces de ese mismo PC. Queremos medir cuándo enviamos el paquete y cuándo llega al destino, y la mejor forma es hacer ambas medidas en la misma máquina ya que eso elimina problemas de sincronización de relojes. Si enviáramos desde un PC y el destino final donde medimos la llegada fuera otro PC, un pequeño desfase en los relojes de las dos máquinas haría muy complicado medir diferencias de tiempos como los que se van a dar en un escenario de LAN, como mucho de unos pocos milisegundos.

2. Tiempos en un enlace

Escoja uno de los PCs (A, B o C). Interconecte su interfaz `eth0` con el interfaz `eth1` (Figura 1) empleando un cable cruzado (los interfaces de esos PCs no implementan Auto-MDI/MDI-X). Active ambos interfaces. Lance el programa Wireshark y póngalo a capturar de ambos interfaces simultáneamente (en la ventana donde selecciona interfaz puede seleccionar más de uno a la vez).

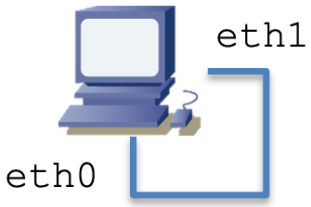

Figura 1 – Dos interfaces del mismo PC interconectados

Emplee `ethSend` para enviar paquetes por el interfaz `eth0`. Haga un envío sin fin de paquetes del tamaño mínimo con separación de 5s entre ellos. Debería aparecer en Wireshark cada paquete dos veces, una de ellas al verlo por el interfaz `eth0` cuando se envía y otra al recibirlo en el interfaz `eth1`.

En Wireshark puede mostrar la diferencia de tiempo entre un paquete y el anterior. Busque la opción en el menú `View > Time Display Format`.

Deberían verse tiempos pequeños entre el paquete enviado y el recibido, seguidos por el tiempo de 5s hasta el siguiente envío. En ocasiones podrían aparecer diferencias de tiempo negativas por la forma en que Wireshark haya leído de los dos interfaces. Las marcas de tiempos las está poniendo la NIC o el sistema operativo, pero wireshark está viendo los paquetes de ambos interfaces alternando entre ellos y puede suceder que con diferencias de tiempos tan pequeñas vea antes el paquete recibido que el enviado, con lo que los muestra en orden contrario al que sucedieron los eventos y al hacer diferencias de tiempos de paquetes consecutivos le sale un tiempo negativo. Nos quedaremos con el valor absolutos de estas diferencias de tiempo.

La herramienta `tshark` es similar a Wireshark, pero su uso es enteramente en línea de comandos (no tiene interfaz gráfico). Pruebe en otro terminal mientras `ethSend` está enviando, por ejemplo:

```
$ tshark -i eth0
1 0.000000000 D-Link_cd:70:4c  -> Broadcast    0x88b5 60 Ethernet II
2 0.000108852 D-Link_cd:70:4c  -> Broadcast    0x88b5 60 Ethernet II
3 5.000078196 D-Link_cd:70:4c  -> Broadcast    0x88b5 60 Ethernet II
4 5.000192072 D-Link_cd:70:4c  -> Broadcast    0x88b5 60 Ethernet II
^C
```

`tshark` dispone de opciones con las que puede controlar la información que muestra para cada paquete. Por ejemplo, con las siguientes opciones leerá de los dos interfaces y mostrará un contador del número de paquete capturado y la diferencia de tiempo con el anterior:

```
$ tshark -i eth0 -i eth1 -l -T fields -e frame.number -e
frame.time_delta_displayed
1       0.000000000
2       0.000108852
3       4.999968773
4       0.000108963
5       4.999968096
6       0.000108641
7       5.000078088
8       -0.000108714
9       5.000078259
10      0.000108697
```

Lance el comando `tshark` con las opciones anteriores, redireccionando esta vez la salida a un fichero de nombre `captura.txt`. Puede validar que `tshark` captura correctamente lanzando desde una tercera terminal el comando `tail` con la opción `-f` (*follow*). Este leerá en tiempo real del fichero de texto e imprimirá en pantalla las líneas nuevas según `tshark` las escriba. Debería ver algo similar al ejemplo anterior.

```
$ tail –f captura.txt
```

En este punto debería tener en un terminal `ethSend` enviando paquetes y, en otro terminal, `tshark` capturando y mandando líneas de texto a un fichero de texto.

Como puede observar, en este fichero de texto se está almacenando la información necesaria para medir los retardos de los paquetes que está enviando. No obstante, también se pueden ver las diferencias de tiempos de ~5s entre paquetes consecutivos y algún caso de diferencias de tiempos negativas, causadas por el orden en el que `tshark` haya leído los paquetes de las distintas interfaces.

3. Procesado de la captura

A continuación, se pide hacer un pequeño programa en Java que permita extraer del fichero de texto anterior los retardos que se quieren medir. Para ello, el programa deberá transformar y filtrar un poco los datos que nos proporciona `tshark`.

Este programa deberá realizar lo siguiente:

- Leer de entrada estándar las líneas generadas por `tshark`.
- Escalar las diferencias de tiempo a milisegundos, ya que es una unidad más cercana a los valores que se van a medir.
- Convertir las diferencias de tiempos a valor absoluto, de forma que se normalicen los tiempos negativos causados por el orden en el que `tshark` lea los paquetes.
- Filtrar aquellos paquetes cuya diferencia de tiempo sea superior a un umbral que indicaremos como argumento. El objetivo será establecerlo en un valor ligeramente inferior al tiempo configurado en `ethSend`, de forma que se dejen de ver los tiempos que no corresponden al tiempo entre envío y recepción.
- Imprimir los tiempos restantes por salida estándar, manteniendo el número del paquete original y la diferencia de tiempo en milisegundos, separados por un tabulador.

En java, para leer texto de la entrada estándar puede utilizar la clase `Scanner`[1]. Se proporciona el siguiente fragmento de código para que si quiere lo utilice como punto de partida del programa a realizar:

```
import java.util.Scanner;
public class Procesado {
	public static void main(String args[]) {
		Scanner sc = new Scanner(System.in);
		while (sc.hasNextLine()) {
			String line = sc.nextLine();
			System.out.printf("%d\t%.6f\n", 1, 0.100000f);
		}
	}
}
```

Tal y como está, si guarda este código en un fichero llamado `Procesado.java`, debería poder compilar y ejecutar el programa con las siguientes instrucciones:

```
$ javac Procesado.java
$ java Procesado < captura.txt
```

[1] https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html

Para proporcionarle a su programa las líneas en tiempo real puede utilizar el comando `tail -f` junto con un *pipe (*ésta será la forma en la que terminaremos haciendo uso del programa). Por ejemplo:

```
$ javac Procesado.java
$ tail -f captura.txt | java Procesado 4000
1       0.000000
2       0.108852
4       0.108963
6       0.108641
8       0.108714
10      0.108697
```

Nota: *Tenga en cuenta qué como `tail -f` se queda a la espera de nuevas líneas, no se cerrará automáticamente la entrada de datos de su programa. Por lo tanto se quedará también a la espera y no terminará a no ser que interrumpa su ejecución manualmente (`^C`).*

4. Visualización de los retardos

Una vez el programa esté funcionando adecuadamente, se van a unir las distintas piezas necesarias para obtener una visualización en tiempo real de los retardos de los paquetes.

1) Deberá tener `ethSend` realizando un envío sin fin de paquetes, **esta vez con un tiempo de 500ms entre ellos** (para evitar tener que esperar 5s entre actualizaciones).

2) Tendrá una instancia de `tshark` capturando en ambos interfaces con las opciones indicadas anteriormente, y **escribiendo el resultado a un fichero de texto `captura.txt`**.

3) El programa del apartado anterior deberá estar en ejecución, **esta vez con un umbral de 400ms**, leyendo los datos que se escriban a `captura.txt` y **volcando el resultado a un fichero de texto `procesado.txt`.**

En otro terminal lance el programa `gnuplot`:

```
$ gnuplot
```

Este programa es interactivo y con él puede hacer una gráfica a partir de un fichero de texto con los datos en columnas simplemente con:

```
gnuplot> plot "procesado.txt" using 1:2
```

La parte de `"using 1:2"` indica que la columna 1 tiene los datos de abscisas y la columna 2 los datos de ordenadas para la gráfica. Obtendrá una figura similar a la de la figura 2.

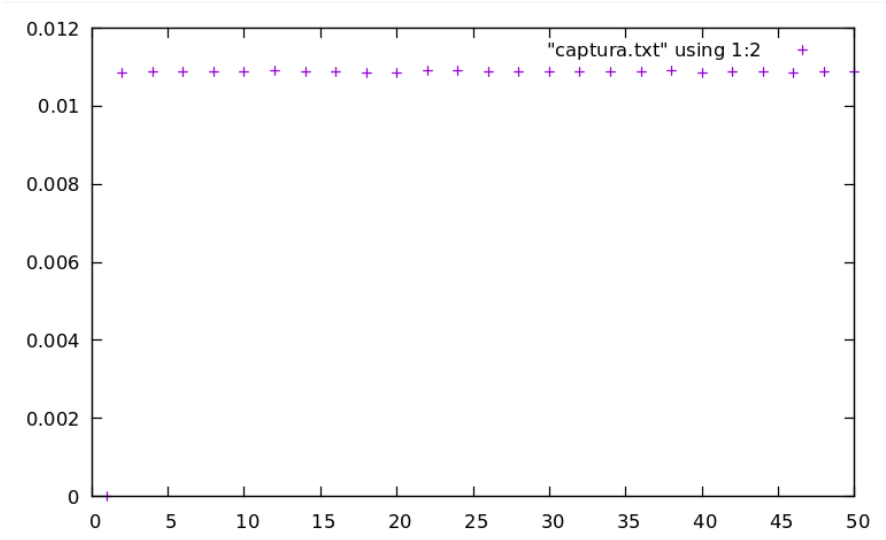

Figura 2 - Gráfica obtenida con `gnuplot`.

Como ese fichero está creciendo a medida que se capturan más paquetes vamos a automatizar que `gnuplot` rehaga la figura cada segundo. Para ello, en la *shell* interactiva que nos ofrece `gnuplot` vamos a ejecutar lo siguiente:

```
gnuplot> while (1) {
more> plot "< tail -n 100 procesado.txt" using 1:2
more> set yrange [0:*]
more> pause 1
more> }
```

Como puede intuir por el código, es un bucle infinito (sí, podríamos decir que gnuplot es programable) donde se dibuja la gráfica, hace una pausa de un segundo y repite. El dibujo de la gráfica se hace con la instrucción `plot`, donde se indica que lea de entrada estándar el resultado de ejecutar `tail -n 100` sobre el fichero `procesado.txt` (se limita a representar las últimas 100 líneas) (sí, gnuplot puede no solo leer ficheros de datos sino ejecutar programas que le generen los datos). Adicionalmente, se está configurando la escala del eje Y para que el siempre empiece en 0 y el límite superior se ajuste automáticamente según los valores encontrados en los datos proporcionados.

Con esto debería tener una gráfica que se va actualizando a medida que aparecen nuevos datos en el fichero. Deberá usar esta visualización para los experimentos del resto de la práctica.

Nota: *Si en algún momento borra o sobre-escribe alguno de los ficheros (por ejemplo, parando y volviendo a lanzar `tshark` o su programa de Java) es probable que tenga que reiniciar de nuevo los pasos posteriores, incluyendo el bucle de dibujado en `gnuplot`.*

Punto de control 1 (30%): Muestre la gráfica obtenida y explique el flujo completo de los datos.

5.1. Enlace directo

En este punto se dispone de las herramientas necesarias para llevar a cabo las medidas con cierta comodidad. Para empezar, se medirán los retardos en un escenario de enlace directo entre ambos interfaces (Figura 1).

Calcule teóricamente cuál es el tiempo de transmisión en los siguientes casos:

- Paquete de 64 bytes, enlace a 100Mb/s.
- Paquete de 1024bytes, enlace a 100Mb/s.
- Paquete de 64 bytes, enlace a 10Mb/s.
- Paquete de 1024bytes, enlace a 10Mb/s.

Mida experimentalmente el retardo de los casos anteriores y compare los valores obtenidos con los teóricos.

Puede indicar el tamaño de los paquetes con `ethSend`. Busque en la ayuda del comando cómo hacerlo.

Puede modificar la velocidad de un interfaz con `ethtool` (si la auto negociación está activada el otro extremo debería adecuarse a ella). Por ejemplo, para establecerla en 10Mb/s:

```
$ sudo ethtool -s eth0 speed 10 duplex full
```

Para reestablecer el interfaz a 100Mb/s es posible que tenga que configurar el interfaz para que anuncie todas las velocidades que soporta:

```
$ sudo ethtool –s eth0 advertise 0x00f
```

Si le interesa, puede ver que significa el `0x00f` con `man ethtool`.

Punto de control 2 (20%): Muestre mediante la gráfica los valores obtenidos en los distintos casos mencionados.

5.2. Store & forward en un conmutador

Interponga un conmutador entre los dos interfaces del PC que está empleando. Es decir, construya algo similar a la Figura 3.

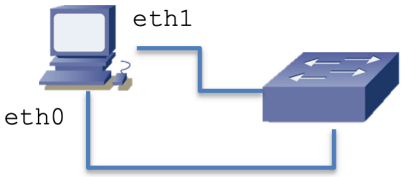

Figura 3 – 2 interfaces del mismo PC al mismo switch

Calcule teóricamente cuál es el tiempo en la red en los siguientes casos:

- Paquete de 64 bytes, ambos enlaces a 100Mb/s.
- Paquete de 1024bytes, ambos enlaces a 100Mb/s.
- Paquete de 64 bytes, un enlace a 100Mb/s y un enlace a 10Mb/s.
- Paquete de 1024bytes, un enlace a 100Mb/s y un enlace a 10Mb/s.
- Paquete de 64 bytes, ambos enlaces a 10Mb/s.
- Paquete de 1024bytes, ambos enlaces a 10Mb/s.

En el conmutador los primeros 4 puertos de cada grupo (1-4, 9-12, 17-20) negocian la máxima velocidad disponible (10, 100 ó 1000 Mb/s) mientras que los restantes (5-8, 13-17, 21-24) están forzados a 10Mb/s. Tenga en cuenta que las NICs de los PCs son a 10/100 Mb/s. En este escenario puede utilizar `ethtool` para fijar la velocidad de las interfaces a su valor de 10Mb/s manteniendo el cableado o hacer uso de los puertos limitados a 10Mb/s para los casos correspondientes (gracias a la funcionalidad de auto-negociación en los interfaces de los PCs).

Realice las medidas experimentales y compare los resultados.

Punto de control 3 (20%): Muestre mediante la gráfica los valores obtenidos en los distintos casos mencionados.

5.3. Store & forward con dos conmutadores

Finalmente interponga dos conmutadores entre los dos interfaces, de forma que pueda ver el efecto de la tasa de transmisión en cada uno de ellos (Figura 4).

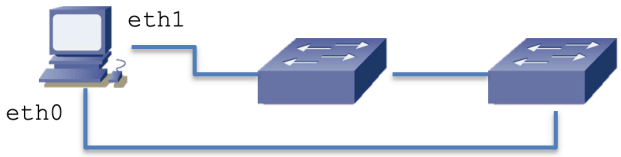

Figura 4 – 2 interfaces del mismo PC con un camino por 2 switches

En este caso, la única forma que tiene de limitar la velocidad del enlace entre los conmutadores es haciendo uso de los puertos limitados a 10Mb/s, mientras que para las interfaces entre el PC y los conmutadores sigue teniendo ambas opciones disponibles.

Realice las pruebas experimentales que considere para verificar que el comportamiento de los retardos son los que espera.

Punto de control 4 (20%): Muestre mediante la gráfica las pruebas realizadas y exponga las conclusiones alcanzadas.

6. Pérdidas de paquetes

En este apartado se pide provocar y ver las pérdidas en el conmutador, dándole más paquetes para reenviar por un puerto de los que pueden salir en un breve periodo de tiempo.

Repita la configuración de la Figura 4. Consiga que ambos enlaces del PC funcionen a 100Mb/s y enlace entre conmutadores a 10Mb/s. Emplee `ethSend` para enviar un flujo continuo de tramas por el interfaz configurado a 100Mb/s. Queremos que estas tramas se reenvíen por el interfaz de 10Mb/s. Como está enviando por un puerto a 100Mb/s puede enviar paquetes a una tasa promedio superior a 10Mb/s. Escoja un tamaño de paquete y un tiempo entre paquetes para conseguir una tasa sostenida que esté por ejemplo entre los 15 y los 20Mb/s.

Mida con Wireshark la tasa promedio del flujo que envía por un interfaz y el que recibe por el otro. Cuente en un intervalo de tiempo la cantidad de paquetes enviados y la cantidad recibida.

Punto de control 5 (10%): Muestre mediante la gráfica la tasa de tráfico que captura en cada interfaz.