Redes de Computadores

Estudio Experimental Sesión de Laboratorio 3.
Nivel de transporte en Internet: UDP y TCP.

Departamento de
Tecnología Electrónica
Última act.: 1/03/2022

Estudio experimental

El estudio experimental de esta práctica consta de dos partes: UDP y TCP. En cada una de ellas se describen todos los pasos que el alumno debe realizar**, si tiene cualquier duda consulte con el profesor encargado de la sesión práctica**. Antes de abandonar el laboratorio debe realizar el punto 55..

(NO ENCIENDA EL PC HASTA QUE SE LO INDIQUEN)

Pasos previos

1. Asegúrese de que está conectado a la red ETSII.
2. Encienda su PC, restaure Windows 7 (**si así se lo indica el profesor**) y arranque Windows 7.
3. Desactive el firewall.
4. Desconecte su PC de la red ETSII y conéctelo a LAB_DTE, concretamente al SWITCH_EUROPA (en G1.31) o al SWITCH_SUDAMERICA (en G1.33). **Si no hubiera puertos disponibles en el SWITCH adecuado indíqueselo al profesor para que le diga lo que hacer**. Compruebe que tiene conectividad de nivel físico.
5. Compruebe que la dirección IP de su PC empieza por 193, anótela, y compruebe que tiene conectividad a nivel de red con su router frontera y con el equipo **webserver.af.lab** (con un ping).
6. No siga adelante hasta que tenga conectividad.
7. Adicionalmente, en esta práctica, haga "clic" en el icono de "Conexión de red"  , luego pulse sobre "Abrir Centro de redes y recursos compartidos" > "Conexión de área local" y seleccione "Propiedades" en el menú contextual. En la ventana que le sale, **desactive** "Cliente para redes Microsoft", y **desactive** "Compartir impresoras y archivos para redes Microsoft" y **desactive** "Programador de paquetes QoS". Pulse "Aceptar" para cerrar la ventana y luego "Cerrar". No deje abierta ninguna ventana.

Primera Parte: Diseñar un filtro para ver sólo las T_PDUs generadas al descargar una pequeña página web.

El objetivo de esta parte de la práctica es poder ver el intercambio de T_PDUs producido al cargar en un navegador la página **http://webserver.af.lab/lab3/paginasimplelab3.html** sin objetos referenciados y con un tamaño muy pequeño. Es el mismo intercambio de T_PDUs visto en el **apartado 2 del estudio teórico**.

La idea principal parece simple, pues bastaría con poner Wireshark a capturar tráfico, arrancar el navegador, cargar la página, detener Wireshark y ver lo que se ha capturado. Sin embargo, no es tan sencillo como parece porque en la captura nos puede aparecer mucho más tráfico del que esperaríamos en un principio, debido a que en nuestro ordenador hay mucho software que genera tráfico de red con el que no contábamos, a que además hay otros dispositivos en la red que también generan tráfico y por último también porque las circunstancias de saturación de la red pueden hacer que se produzcan retransmisiones con las que en un principio tampoco contábamos. Es muy normal o incluso "imposible" que no obtengamos a la primera una captura "limpia" que contenga solo el tráfico que queremos estudiar.

En el caso de que veamos que la captura hecha con Wireshark no sea "limpia", es decir, que no sea la que esperábamos basándonos en nuestros conocimientos teóricos, caben varias soluciones posibles:

A) Intentar configurar el software de nuestro PC de forma que genere la menor cantidad posible de tráfico "indeseado". Eso es lo que hemos hecho, por ejemplo, en el apartado 7..
B) Aplicar filtros de visualización a la captura "imperfecta" de forma que solo se nos muestre el tráfico deseado. Estos **filtros** pueden preparase **antes de hacer la captura**, pero también es útil prepararlos **después de haber hecho la captura**, una vez que hayamos obtenido los valores adecuados para ciertos parámetros del filtro (por ejemplo números de puerto, direcciones IP, etc.).
C) Repetir de nuevo la generación de tráfico y la captura del mismo hasta que la fortuna haga que obtengamos una captura "limpia", con el tráfico deseado.

Nosotros vamos a utilizar las tres técnicas: Primero, configurar el PC para que genere poco tráfico "extra". Segundo, usar filtros de visualización que nos eviten ver la mayoría del tráfico que no nos interesa y que sepamos que de no filtrarlo nos va a aparecer en la mayoría de las capturas. Por último, si a pesar de los filtros se nos muestra cierto tráfico indeseado pero que sea poco probable que aparezca, es mejor repetir el proceso de generación y captura de tráfico de red en lugar de diseñar un filtro de visualización complejo que nos lo oculte.

8. En la práctica anterior, se nos explicaba la necesidad de configurar el comportamiento del navegador Mozilla Firefox para adaptarlo a la intranet del laboratorio, la cual tiene unos enlaces entre routers con un ancho de banda muy bajo (de unos 56000bps), de forma que el navegador no abra más conexiones TCP de las necesarias, a pesar de que el tiempo de establecimiento de conexión sea largo. **Siga de nuevo los mismos pasos:** Abra Mozilla Firefox, entre en el URL especial **about:config**, acepte el "aviso para manazas", busque la frase "retry-timeout", modifique con doble "clic" el valor de la preferencia "network.http.connection-retry-timeout" para que valga 12000 y cierre Mozilla Firefox.
9. Entre en Wireshark y asegúrese de que las opciones **"Resolve network (IP) addresses"** y **"Resolve transport names"** están **desactivadas**.
10. **Ponga a Wireshark a capturar tráfico y no lo detenga hasta que se le indique expresamente.** Estará viendo que se recibe al menos una trama nueva cada dos segundos. Esto indica que va a ser imposible hacer una captura "limpia" en la red del laboratorio, pues sin haber generado el tráfico de red que nos interesa, ya estamos capturando tramas. **Necesitaremos usar filtros de visualización** para ver lo único que nos interesa: las T_PDUs relacionadas con la descarga de una página web.
11. Como la página web que estamos interesados en capturar está en el servidor web llamado **webserver.af.lab** se generará tráfico DNS para preguntar por ese nombre. Además sabemos que la conexión TCP se efectuará al puerto 80 del servidor web, que es el **"puerto bien conocido"** que usa HTTP. Por tanto escriba **((dns.qry.type==1 and dns contains webserver) or tcp.port==80) and ip.addr==193.1.X.Y** (debe sustituir 193.1.X.Y por la dirección IP de su PC) como filtro de visualización en Wireshark, compruebe bien que es correcto y aplíquelo. Con el filtro **dns.qry.type==1** estamos visualizando únicamente cierto tipo de tráfico DNS, el que se genera al preguntar por la **dirección IP versión 4** de un cierto host. Probablemente ese filtro no nos esté mostrando ahora ninguna trama, pero pronto veremos que es capaz de mostrarnos todas las tramas que nos interesan. Anote los valores de los contadores "Packets" y "Displayed" y avise al profesor si "Displayed" no es 0.

12. Un filtro aún mejor, si supiésemos cuál es la dirección IP del servidor web, sería el siguiente (A.B.C.D es la dirección IP del servidor web): **((dns.qry.type==1 and dns contains webserver) or (tcp.port==80 and ip.addr==A.B.C.D)) and ip.addr==193.1.X.Y**. Si más adelante, tras capturar el tráfico, resultase que el filtro del apartado 11. no es satisfactorio, aprovechando que la captura nos permite ver cuál es la IP del servidor web, podríamos cambiar el filtro del apartado 11. por este otro.

Segunda Parte: Analizar el intercambio de T_PDUs generadas al descargar una pequeña página web.

Por fin vamos a generar el tráfico de red que queremos analizar. Siga las siguientes instrucciones.

13. La caché DNS de Windows evita que se haga un uso excesivo de tráfico DNS. En nuestro caso, como queremos asegurarnos de que se genera tráfico DNS, lo que vamos a hacer es borrarla antes de empezar a generar tráfico. **Borre la caché DNS de Windows** usando el comando adecuado en una ventana de Símbolo del sistema. Recuerde que utilizó este comando de borrado en la práctica anterior. **Anote el comando de borrado de caché DNS.**
14. **Abra el navegador Mozilla Firefox y borre su caché de páginas.** Recuerde que en el estudio experimental de la práctica anterior se le enseñó cómo hacerlo. **Anote** cuál es el **procedimiento de borrado de la caché de páginas del navegador**.

15. **Si Wireshark estaba capturando y NO se ven tramas en el listado de tramas**, entonces no es necesario hacer nada. **Si Wireshark estaba capturando y se ven tramas en el listado de tramas**, entonces debemos pulsar el icono "RESTART" para descartar todo lo capturado y empezar a capturar desde cero. **Si Wireshark no estaba capturando,** entonces hay que ponerlo a capturar.
16. Asegúrese de que la ventana de Mozilla Firefox no tiene un tamaño demasiado grande, de forma que pueda ver detrás la ventana de Wireshark lo que está ocurriendo en el listado de tramas. Escriba en el navegador el URL **http://webserver.af.lab/lab3/paginasimplelab3.html** para cargar la página web y generar tráfico en la red.
17. **Espere unos 30 segundos** a que pare el tráfico en la red y compruebe el listado de tramas y el número que se le muestra en el contador "Displayed". **Pueden pasar dos cosas:**
    - **A)** Que se estén visualizando **exactamente 13 tramas** y una de ellas muestre en la columna "Info" el texto **"HTTP/1.1 200 OK"**. Eso indica que todo ha ido bien, así que **detenga la captura** de Wireshark y **pase ya al siguiente apartado**, el 18..
    - **B)** Que no ocurra exactamente lo que indica el punto A) anterior. Eso es porque algo ha ido mal y debe **cerrar el navegador Mozilla Firefox (MUY IMPORTANTE) y repetir prestando mucha atención todos los pasos desde el apartado** 13. **hasta éste**. Si ha repetido el proceso, sin éxito, varias veces, revise que el filtro del apartado 11. esté bien escrito y haya sido aplicado correcta-

mente. Puede probar a usar el filtro mejorado del apartado 12. o avisar a su profesor para que le oriente. **No pase al apartado** 18. **sin cumplir las condiciones del punto** A)**.**

18. En Wireshark, pulse el icono **"SAVE"** (o bien entre en **"File > Save"**) para guardar todas las tramas capturadas (aunque no se estén visualizando). Use **"todaslastramas"** como nombre de fichero de captura.
19. Wireshark también permite grabar en un fichero sólo una parte de las tramas capturadas. En concreto es muy útil grabar solo las tramas que nos muestra el filtro de visualización actual. Vamos a guardar las 13 tramas visualizadas entrando en el **"File" > "Export Specified Packets..."** y marcando dentro del apartado llamado "Packet Range" las casillas "All packets" y "Displayed" (si no estaban ya marcadas). Use el nombre **"13tramas"** para el archivo de captura.
20. Desactive el filtro de visualización pulsando en el **botón "Clear"** y **anote** el valor del contador **"Displayed"** cuando no se usa filtro. Escriba y aplique (por separado) los **cuatro filtros** simples **dns**, **udp**, **tcp** y **http** y anote el valor del contador **"Displayed"** para cada uno de ellos. Esto le permite hacerse una idea de la potencia del filtro complejo utilizado y que solo nos mostraba 13 tramas.

21. Desactive de nuevo el filtro de visualización pulsando en el **botón "Clear".** Para poder trabajar sin usar un filtro vamos a **cargar el archivo que solo contiene 13 tramas**. Para ello, pulse el icono de la carpeta y **cargue el fichero** que grabó antes con nombre **"13tramas"**.
22. **Fíjese en las 13 tramas** que Wireshark le está mostrando en el listado de tramas. Esas son las tramas **generadas como consecuencia de la carga de la página web simple** en el navegador. Las dos primeras tramas están relacionadas con el **tráfico DNS** que ha tenido lugar con el servidor DNS con objeto de resolver el nombre del servidor web **webserver.af.lab**. Las 11 tramas restantes están relacionadas con el **tráfico HTTP** necesario para pedirle al servidor web la página.
23. Escriba y aplique el filtro **dns** para ver solo las tramas que encapsulan DNS_PDUs.
    ¿Cuántas peticiones DNS hay relativas a la carga de la página web del punto 16.?
    ¿Qué IP tiene el servidor DNS?
    ¿Qué IP tiene el equipo de nombre **webserver.af.lab**?
    ¿Qué RTT se observa para la respuesta a la petición?
    Nota: si para ver el RTT usa la técnica de marcar una trama como referencia de tiempo "Set Time Reference" no olvide hacer desaparecer la marca ***REF*** antes de pasar al siguiente apartado.
24. Escriba y aplique el filtro **udp** para ver solo las tramas que encapsulan UDP_PDUs.
    ¿Por qué el filtro **dns** y el filtro **udp** muestran exactamente las mismas tramas?
    ¿En qué casos los filtros **dns** y **udp** podrían mostrar cosas diferentes?

    ¿Cuántas T_PDUs del protocolo UDP se han intercambiado relativas a la carga de la página simple?
    ¿Coincide esto con la respuesta que ha dado al **apartado 2.a del estudio teórico**?
    ¿Cuántos bytes ocupa la UDP_UD de cada una de esas UDP_PDUs?
    Nota: No cometa el error de usar la columna "Length" para responder a esta última pregunta. Use el panel de detalles de trama y recuerde cuál es la estructura, significado y tamaño de los campos de la UDP_PCI presente en la UDP_PDU.
25. Escriba y aplique el filtro **http** para ver solo las tramas que encapsulan HTTP_PDUs. Es decir, para ver exclusivamente el intercambio de A_PDUs entre las dos entidades de aplicación (el navegador y servidor web). Fíjese en la información del listado de tramas y responda.
    ¿Cuántas A_PDUs hay del protocolo HTTP relativas a la carga de la página web del punto 16.?
    ¿Cuántas peticiones HTTP hay?
    ¿Qué IP tiene el servidor web de nombre **webserver.af.lab**?
    ¿Qué IP tiene nuestro PC?
    ¿Qué RTT se observa entre una petición HTTP y su correspondiente respuesta?
    Nota: si para ver el RTT usa la técnica de marcar una trama como referencia de tiempo "Set Time Reference" no olvide hacer desaparecer la marca ***REF*** antes de pasar al siguiente apartado.
26. Escriba y aplique el filtro **tcp** para ver solo las tramas que encapsulan TCP_PDUs. Esto nos va a permitir ver el intercambio de TCP_PDUs entre la entidad TCP de nuestro PC y la entidad TCP del equipo en el que está el servidor web.
    ¿Cuántas T_PDUs del protocolo TCP se han intercambiado relativas a la carga de la página?
    ¿Coincide esto con la respuesta que ha dado al **apartado 2.b del estudio teórico**?
    ¿Por qué dos tramas del listado muestran **HTTP** en la columna **"Protocol"** en lugar de **TCP**?
    ¿Hay también una TCP_PDU dentro de esas dos tramas?
    El intercambio de TCP_PDUs es mucho más numeroso que el intercambio de HTTP_PDUs, según se observa en el listado de tramas.
    ¿Cuántas TCP_PDUs hay que no encapsulan datos del nivel de aplicación?
    ¿Para qué sirven las TCP_PDUs que tienen una TCP_UD de tamaño cero?

Avise al profesor por si quiere revisar las respuestas de los cuatro apartados anteriores.

27. Observe que las dos primeras TCP_PDUs del listado de tramas aparecen marcadas en la columna "Info" con las etiquetas **[SYN]** y **[SYN, ACK]** respectivamente.
    ¿En qué se basa Wireshark para etiquetar esos dos segmentos con la palabra **SYN**?
    ¿Es por el hecho de ser los dos primeros segmentos del fichero de captura?
    ¿Es porque hay algún dato especial en el interior de esos dos segmentos?
    Si es así, localice ese dato en el panel central (el panel de detalles de trama).

28. Lo mismo pasa con los segmentos etiquetados con [**FIN**]. Localice en el panel central ese dato especial que hace diferentes a los segmentos que muestran la palabra **FIN** en el campo "Info".

29. **Realice un diagrama temporal donde aparezcan los 11 segmentos TCP** representados como flechas que van de izquierda a derecha o de derecha a izquierda con una cierta inclinación. El tiempo, transcurre de arriba abajo. Etiquete cada segmento con el Nº de Secuencia (**Seq=**), Nº de reconocimiento (**Ack=**), y tamaño de los datos de nivel superior transportados dentro del segmento (**Len=**). Si el segmento tiene activados bits especiales como por ejemplo el bit **SYN** o **FIN**, indíquelo también. Todos los datos que necesita para hacer el gráfico los tiene a la vista en lo que se muestra en la columna "Info" para cada segmento TCP. Solo encontrará algo de dificultad en los segmentos TCP que contienen datos del protocolo HTTP, porque entonces la columna "Info" en lugar de mostrar los detalles de TCP muestra los detalles de HTTP. Para resolver este problema, de esos segmentos TCP puede consultar el **Seq**, **Ack** y **Len** (y otros detalles de TCP) haciendo "clic" sobre la trama que los contiene y mirando la línea "Transmission Control Protocol" en el panel central (detalles de trama), que contiene información resumida muy útil.
30. Compare el diagrama temporal que ha realizado ahora con el diagrama temporal que realizó en el **apartado 2.b del estudio teórico**. Deben ser muy similares. ¿Hay alguna diferencia destacable?

Avise al profesor por si quiere revisar el diagrama temporal que ha realizado.

31. Si sólo estamos interesados en ver en el listado de tramas todos los detalles del protocolo TCP de forma cómoda y no estamos interesados en ver el protocolo HTTP, podemos decirle a Wireshark que nos **oculte los detalles de HTTP** totalmente. Hágalo: Entre en "Analize > Enabled Protocols", haga "clic" en la lista de protocolos, teclee HTTP para localizar rápidamente ese protocolo en la lista y luego **desmarque** su casilla en la columna "Status" y pulse "OK". Verá como ahora las 11 tramas que contienen segmentos TCP (incluso las dos que contienen datos HTTP) muestran solo detalles TCP.
32. ¿Qué tamaño tiene la A_PDU enviada por el cliente?
33. ¿Qué tamaño tiene la A_PDU enviada por el servidor?
34. En el gráfico que ha elaborado en el punto 29., tanto el servidor como el cliente parece que están usando como número de secuencia inicial el cero. Eso no es cierto realmente. Lo que ocurre es que por defecto, al analizar el protocolo TCP, Wireshark lo que nos muestra son **números de secuencia relativos**. Es decir, que en lugar de mostrarnos los **números de secuencia reales** del cliente, a esos números les resta el valor inicial usado por el cliente, de forma que nosotros lo que vemos es siempre un numero de secuencia cero en el primer segmento, el segmento SYN, y luego van incrementándose.
35. Entre en "Edit" > "Preferences" y dentro de la rama "Protocols" busque TCP (tecleando TCP) y **desmarque** la casilla que llamada **"Relative sequence numbers"**. Pulse "OK" salir. Observe como la columna "Info" del listado de tramas muestra ahora los **números de secuencia reales** enormes (son enteros positivos de 32 bits) a los que es muy complicado seguirles la pista.
36. Fíjese bien en los dos primeros segmentos TCP. ¿Son iguales los números de secuencia iniciales escogidos por el cliente y por el servidor?
37. ¿En qué segmento del listado vemos que envía el servidor su primer ACK?
38. ¿Qué diferencia hay entre el número de secuencia inicial del cliente y el primer número de reconocimiento (ACK) enviado por el servidor?
39. Observe que el cliente **no** envía ACK en el primer segmento TCP que envía. El cliente aún no conoce el número de secuencia inicial del servidor, así que esta es la única ocasión en la que un segmento TCP no lleva el bit ACK activado (a valor 1). Compruebe esto escribiendo y aplicando el filtro **tcp.flags.ack==0** que nos mostrará solo un segmento TCP, aquel que tiene el bit ACK a valor 0.
40. El término **flags** es el que usa Wireshark para referirse a los **bits** que aparecen en la TCP_PCI y que nosotros llamamos bit ACK, bit SYN, bit FIN y bit RST de la TCP_PCI. Es frecuente el uso de filtros como **tcp.flags.syn==1** o como **tcp.flags.fin==1** para estudiar con Wireshark el establecimiento y cierre la conexión TCP.
41. Aplique de nuevo el filtro **tcp** en Wireshark. ¿Qué diferencia hay entre el número de secuencia inicial del servidor web y el primer número de ACK enviado por el cliente?

42. Deshaga los cambios que ha hecho en la configuración de Wireshark: Es decir, **active los números de secuencia relativos** en TCP y **active** de nuevo el análisis del protocolo **HTTP**. Esto es importante porque en los apartados siguientes **nos referiremos siempre a números de secuencia relativos**.
43. En el punto 32. anotó el tamaño de la única A_PDU enviada por la aplicación cliente. Cada octeto de esa A_PDU tiene asignado un número de secuencia dentro del flujo de datos que van del cliente al servidor por la conexión TCP. El cliente ha usado el número de secuencia 0 para asignárselo al bit SYN. A partir de ahí ha empezado a asignarle números de secuencia a los octetos de la A_PDU y por último, ha asignado un numero de secuencia al bit FIN.
    ¿Qué número de secuencia es el que ha asignado la entidad TCP cliente a su bit FIN?
44. ¿Qué número de secuencia ha asignado la entidad TCP del servidor a su bit FIN?

45. ¿Es posible saber cuál de las dos aplicaciones (la cliente o la servidora) ha enviado más datos a través de la conexión TCP simplemente viendo el número de secuencia (relativo) que asignan las entidades TCP cliente y servidora a su propio bit FIN en el momento de enviarlo?
    Si es así, ¿qué relación existe entre el número de secuencia (relativo) del bit FIN y la cantidad de datos de nivel de aplicación enviados a través de una conexión TCP?

46. Cuando TCP envía un segmento sin datos, está obligado a ponerle un número de secuencia al segmento. Ese número de secuencia debe ser el número de secuencia que tocaría utilizar si se fuese enviar un segmento conteniendo datos "nuevos" no enviados antes. Puede comprobar esto en cualquiera de los segmentos con **Len=0** pero quizá sea especialmente interesante hacerlo en el último segmento TCP mostrado en el listado y enviado por el servidor. Fíjese como en ese segmento sin datos el número de secuencia usado es el número siguiente al que se usó para asignárselo al bit FIN del servidor.
47. ¿Qué hace la entidad TCP cuando la entidad de aplicación a la cual presta servicio le da la orden de cerrar la conexión TCP?
48. ¿Qué entidad de aplicación es la primera que ha decidido cerrar la conexión?
49. Fíjese en el listado de tramas en que cuando una de las entidades TCP ha enviado un segmento FIN, la otra entidad TCP también envía un FIN ¿Lo hace casi inmediatamente o tarda algunos segundos?

50. Observe la HTTP_PCI de la HTTP_PDU enviada por la aplicación servidora. ¿Qué relación existe entre el instante de tiempo preciso en el que cierra la conexión el servidor y esa HTTP_PCI?

51. **Anote** el tiempo transcurrido entre que el cliente envía el SYN y recibe el SYN-ACK. Eso le puede dar una idea del **RTT entre cliente y servidor** (tiempo de ida y vuelta) cuando se envía una T_PDU vacía y vuelve una T_PDU vacía.
52. **Anote** también el **RTT** observado entre cliente y servidor desde que el **cliente envía el FIN** y hasta que el cliente **recibe el ACK de ese FIN**. ¿Es similar al del apartado anterior?
53. **Anote** ahora el **RTT** que se aprecia **entre el GET enviado** por el cliente **y el ACK recibido**.
    ¿Es mayor o menor que los dos RTT que ha calculado anteriormente?
    ¿Por qué cree que ocurre eso?

54. El tercer segmento TCP mostrado en el listado se trata de un segmento de ACK. Este segmento de ACK sirve para reconocerle al servidor la llegada del segmento SYN-ACK. Piense qué habría pasado si este ACK se pierde por el camino y no llega al servidor. Fíjese en que detrás de ese segmento, el cliente TCP envía un segmento conteniendo datos de aplicación.
    ¿Serviría ese segmento para reconocerle al servidor la llegada del segmento SYN?
    Si es así…¿Cree que llegaría ese ACK a tiempo o demasiado tarde?
    Nota: Fíjese en el instante en que se envía el tercer segmento y compare con el instante en que se envían los datos de aplicación contenidos en el cuarto segmento.

    Avise al profesor por si éste quisiera revisar el experimental o hacerle alguna pregunta.

55. Antes de irse debe dejar el PC apagado y conectado a la red ETSII (como estaba al principio) y también debe devolver a su sitio el latiguillo que ha usado para conectarse a la red LAB_DTE.