UNIVERSIDAD DEL VALLE DE GUATEMALA
CC3067 – REDES
Sección 10

LABORATORIO 1

Red Humana e Introducción a Wireshark

Michelle Angel de María Mejía Villela, 22596

Guatemala, 17 de julio del 2025

Contenido

Link del repositorio: https://github.com/michellemej22596/RedesLab1

Antecedentes

Previo a las redes de computadoras, como las conocemos hoy en día, la transmisión de información era de persona a persona. Sus primeros y más grandes usos fueron el telégrafo y el teléfono. Más adelante, con el objetivo de mejorar la tasa de transmisión, se aprovecharon sistemas automatizados que pudieran no solo controlar las rutas para llamadas telefónicas o la comunicación de datos digitales.

Por otra parte, es importante analizar y entender los paquetes que se transmiten durante la comunicación entre dispositivos. Los analizadores de red como Wireshark ayudan a realizar esta tarea y son una de las principales herramientas utilizadas por los administradores de redes, hackers éticos, etc.

Es por ello que realizaremos ciertas actividades introductorias a la ciencia de enviar información que nos permitirán percatarnos de muchos detalles, y también aprenderemos a configurar y utilizar Wireshark y sumarnos a la comunidad de científicos de computación, administradores y analistas de red, hackers, etc., que la utilizan, con el fin de asegurar el óptimo rendimiento y seguridad de nuestras redes.

Objetivos

- Identificar ventajas y desventajas de distintos esquemas de comunicación.
- Comprender e identificar la complejidad al momento de enviar información.
- Conocer las bases de un conmutador a pequeña escala.
- Conocer los propósitos y usos de una analizador de paquetes
- Familiarizarse con el entorno de Wireshark
- Fortalecer la teoría sobre paquetes a través del análisis de paquetes reales

Desarrollo

El laboratorio cuenta con dos partes principales. La primera parte se debe hacer de forma presencial/sincrónica para contar con sus parejas. La segunda parte puede realizarse en clase, o bien de manera asincrónica/de tarea. Al final, el documento/reporte a entregar lo trabajan y entregan de forma individual.

Red Humana (parte grupal en clase)

Integrantes: Silvia Illescas 22376 y Michelle Mejía 22596

Existen distintas formas en que se puede representar la información al momento de enviarla a través de un medio. Un esquema utilizado en los tiempos de la telegrafía fue el código Morse, el cual podía representar, a través de pulsos, todas las letras del alfabeto en inglés y los 10 dígitos arábigos (ver **Imagen 1**). No obstante, con mejoras en los sistemas de telégrafo y su parcial automatización se introdujeron nuevos sistemas que podían aprovechar los avances tecnológicos. Uno de ellos fue el código de Baudot, el cual podía codificar todas las letras del alfabeto inglés, junto con códigos de control, a través de representaciones binarias de 5 bits (ver **Imagen 2**).

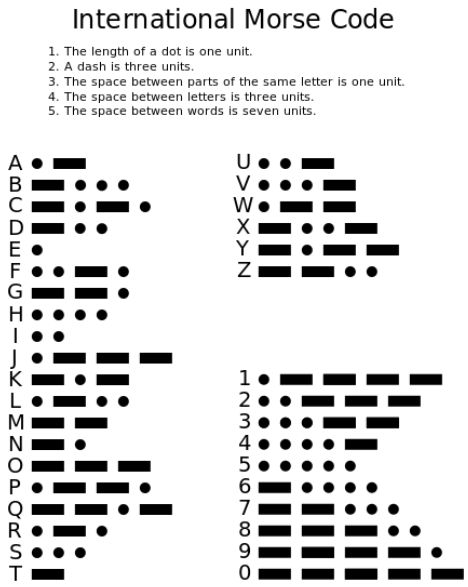


***Imagen 1**: Código Morse internacional.*
Fuente
https://en.wikipedia.org/wiki/File:International_Morse_Code.svg

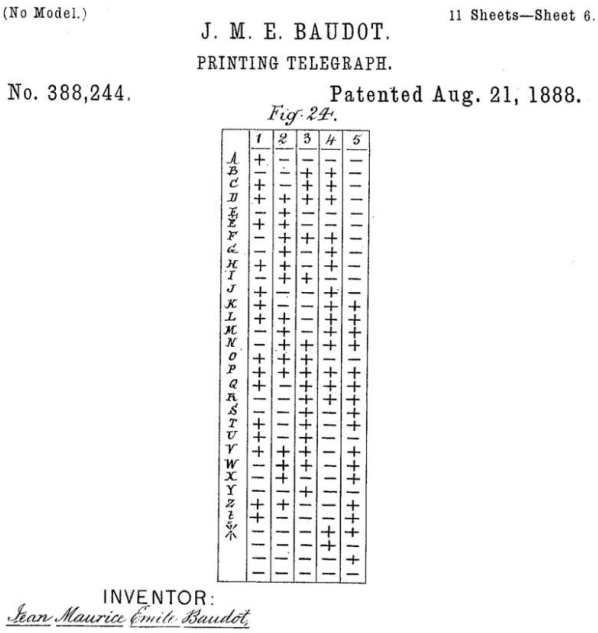


***Imagen 2**: Código de Baudot.*
Fuente
https://en.wikipedia.org/wiki/Baudot_code /media/File:Baudot_Code_-_from_1888_patent.png

Para el laboratorio se estarán enviando estos mensajes a través de un medio. Debido al factor remoto/híbrido estaremos haciendo una versión un poco más “moderna” de la actividad: enviaremos los mensajes emulando con nuestra voz, o con un objeto (i.e.: un lapicero en su escritorio), un generador de audio, etc., los elementos de nuestros mensajes. En el caso del código Morse pulsos (con duración variable) y en el caso del código de Baudot bits (1’s y 0’s).

La forma de envío de los mensajes depende de cada una de las partes de la actividad, como se detalla a continuación...

1.1 Primera parte: transmisión de códigos

En la primera parte se estarán distribuyendo en **parejas** (o un trío en caso de grupo impar). Cada pareja deberá practicar el envío y la recepción de mensajes utilizando los dos esquemas. Intentar enviar **al menos tres mensajes distintos (de 10 caracteres mínimo) por persona**, por cada uno de los esquemas. La comunicación se hará en un **Room de Zoom con su pareja, usando su micrófono** para enviar los mensajes... **En caso presencial, la comunicación se hará directamente** (frente a frente, hablado). Durante la actividad, tengan en mente las siguientes preguntas (las debe incluir en su reporte):

- **¿Qué esquema es más fácil? ¿Más difícil?**

Dependiendo de cómo emitamos el sonido para el código morse, nos pareció bastante rápido la interpretación y sencillo siempre y cuando se distingan bien las rayas de los puntos y la separación entre letras. El código Baudot estuvo bastante bien (no lo clasificaría como difícil) ya que sabíamos la cantidad de dígitos que contenía cada carácter y era más difícil perder información, sin embargo fue un poco más lento el proceso de comunicación y traducción (por lo que nos dejamos guiar por el código morse).

- **¿Con cuál ocurren menos errores?**

Con el Baudot, ya que es más sencillo comunicar los 0s y 1s, además que la longitud ya está establecida. Tuvimos un 100% de éxito con esta forma, únicamente pasaba que, si confundía un bit de vez en cuando, pero lo arreglaba solamente revisando el resto de las letras y dándole sentido a la palabra.

1.2 Segunda parte: transmisión “empaquetada”

En la segunda parte repetiremos la dinámica anterior **utilizando únicamente el esquema que más se les haya facilitado.** En este caso, el envío se hará de una forma diferente: **mediante notas de voz (VN) enviadas por Whatsapp/Discord/etc. donde se graben ustedes emitiendo el mensaje en código**. Deben intentar enviar **al menos tres mensajes (de 10 caracteres) por persona, diferentes a los mensajes anteriores.** Durante la actividad, tengan en mente lo siguiente:

- **¿Qué dificultades involucra el enviar un mensaje de esta forma “empaquetada”?**

Fue muchísimo más complicado ya que cuando lo hicimos conversando esperamos inconscientemente a que la persona que estaba recibiendo el mensaje nos avisará que estaba lista para continuar, si necesitaba repetir algo o simplemente el cambio de carácter a otro. En cambio, por medio de WhatsApp los audios no se distinguían tan bien, y era un poco más complejo saber que ya había cambiado de letra… Si no escuchabas algo podías repetirlo, pero era difícil ver por dónde me había quedado en la nota de voz.

1.3 Tercera parte: conmutación de mensajes

En la tercera parte la clase se repartirá cooperando con otra pareja/grupo, con quienes deberán determinar lo siguiente:

- Tres+ personas serán los clientes del servicio
- Una persona funcionará como conmutador

Entre ustedes se organizarán según la Topología que se muestra en la **Imagen 3**.

Integrantes: Silvia Illescas 22376, Ruth de León 22428, Isabella Miralles 22293 y Michelle Mejía 22596

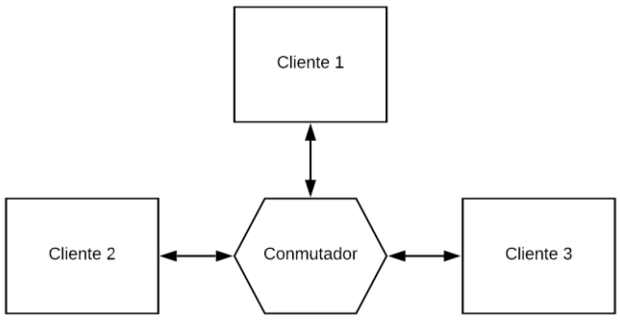


***Imagen 3**: Comunicación entre clientes y conmutador.*

El conmutador recibirá la VN de Whatsapp/Discord de cualquiera de los clientes y luego lo estará reenviando al destino final. Para ello, deben de acordar cómo dirán al conmutador quién es el destino final del mensaje, así como determinar si el conmutador está listo o no para recibir mensajes. Durante la actividad, tengan en mente lo siguiente.

- **¿Qué posibilidades incluye la introducción de un conmutador en el sistema?**
  Tenemos la posibilidad de enviar mensajes a una red más amplia de personas y no solamente uno a uno. Por lo que el alcance es mayor y no es necesario tener el número de todos nuestros compañeros, basta con compartirlo con nuestro conmutador central.

- **¿Qué ventajas/desventajas se tienen al momento de agregar más conmutadores al sistema?**

  Es un poco más complicado saber a cuál de todos los conmutadores solicitar la petición, sin embargo, en ventajas tendremos menor tiempo de espera para enviar un mensaje (ya que estarán más tiempo libre) y podemos cubrir una mayor demanda o bien redundancia en nuestra información enviada.

  Protocolo: Enviamos la primera letra la inicial del cliente al que deseamos que llegue la información, ejemplo, I para Isabella, S para Silvia, R para Ruth. Luego acompañamos la inicial con el mensaje cifrado (código morse: punto/raya). Y finalmente el conmutador (Michelle) se encarga de reenviar el mensaje al cliente adecuado. El conmutador no recibirá mensajes si se encuentra descifrando alguno previo. Puede avisar por medio de un mensaje de confirmación que se encuentra libre ante la petición de un envío de mensaje de un cliente.

Introducción a Wireshark (parte individual de tarea)

Se debe descargar e instalar el software de Wireshark. Es probable que para ejecutarlo pida permisos de administrador (sudo, click + run as admin, etc.).

1.1 Primera parte: personalización del entorno

En la primera parte se realizará la personalización del entorno de Wireshark, de modo que se adapte a nuestras preferencias de uso.

1. Descargue el archivo https://www.cloudshark.org/captures/e6fb36096dbb (Export -> Download)

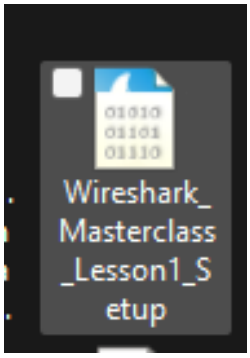


Figura 1. Descarga del archivo

2. Cree un perfil (Configuration -> Profiles) con su primer nombre y primer apellido

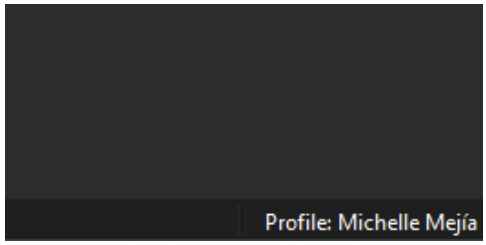


Figura 3. Creación del perfil

3. Abra el archivo descargado (File -> Open)

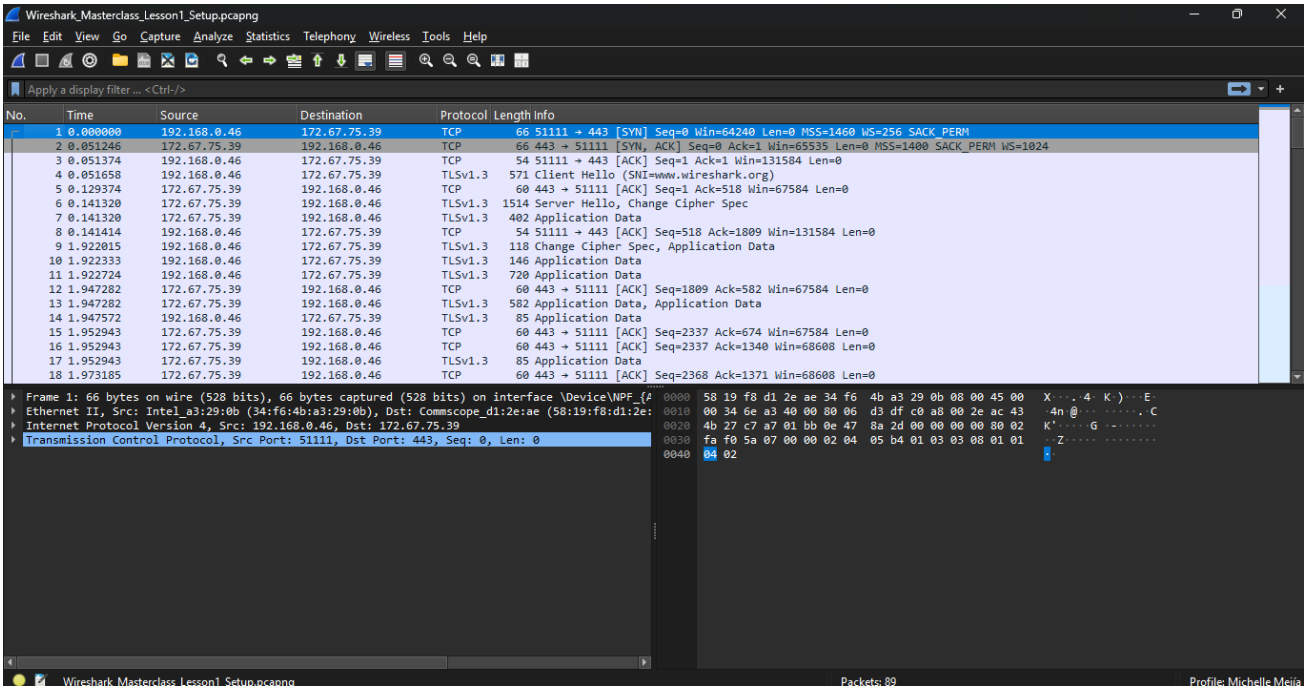


Figura 4. Apertura de archivo

4. Aplique el formato de tiempo Time of Day a la columna Tiempo (View -> Time Display)

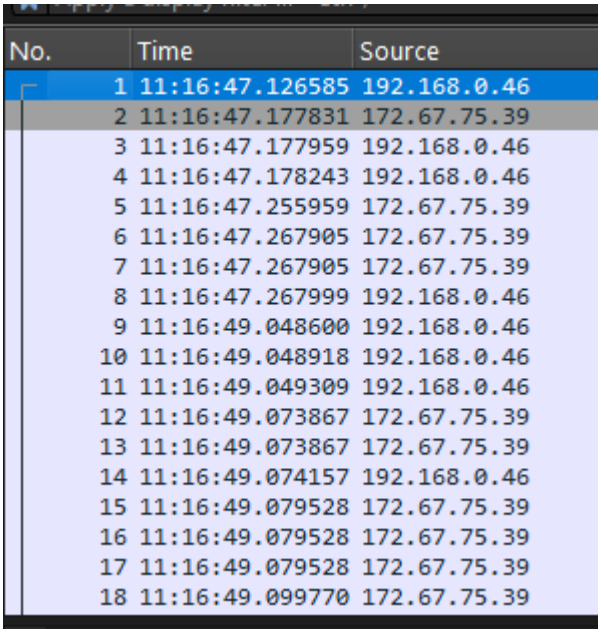
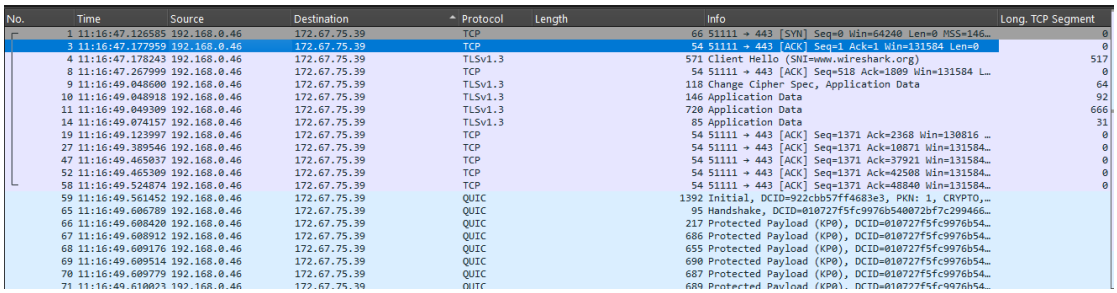


Figura 5. Formato Time of Day

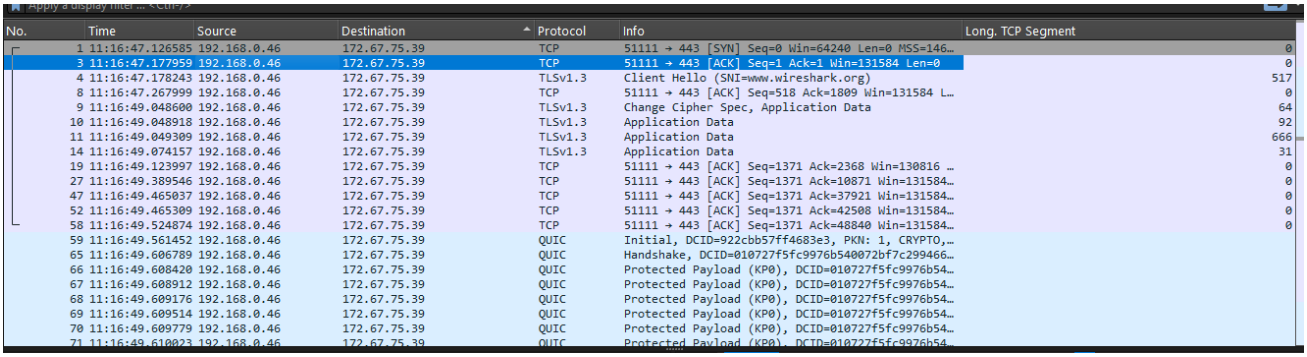
5. Agregue una columna con la longitud del segmento TCP (Selecciona la primera fila, en el panel inferior despliegue Transmission Control Protocol, seleccione TCP Segment Len y apliquelo como una columna)



No.	Time	Source	Destination	Protocol	Length	Info	Long. TCP Segment
1	11:16:47.126585	192.168.0.46	172.67.75.39	TCP		66 51111 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=146…	0
3	11:16:47.177959	192.168.0.46	172.67.75.39	TCP		54 51111 → 443 [ACK] Seq=1 Ack=1 Win=131584 Len=0	0
4	11:16:47.178243	192.168.0.46	172.67.75.39	TLSv1.3		571 Client Hello (SNI=www.wireshark.org)	517
8	11:16:47.267999	192.168.0.46	172.67.75.39	TCP		54 51111 → 443 [ACK] Seq=518 Ack=1809 Win=131584 L…	0
9	11:16:49.048600	192.168.0.46	172.67.75.39	TLSv1.3		118 Change Cipher Spec, Application Data	64
10	11:16:49.048918	192.168.0.46	172.67.75.39	TLSv1.3		146 Application Data	92
11	11:16:49.049309	192.168.0.46	172.67.75.39	TLSv1.3		720 Application Data	666
14	11:16:49.074157	192.168.0.46	172.67.75.39	TLSv1.3		85 Application Data	31
19	11:16:49.123997	192.168.0.46	172.67.75.39	TCP		54 51111 → 443 [ACK] Seq=1371 Ack=2368 Win=130816 …	0
27	11:16:49.389546	192.168.0.46	172.67.75.39	TCP		54 51111 → 443 [ACK] Seq=1371 Ack=10871 Win=131584…	0
47	11:16:49.465037	192.168.0.46	172.67.75.39	TCP		54 51111 → 443 [ACK] Seq=1371 Ack=37921 Win=131584…	0
52	11:16:49.465309	192.168.0.46	172.67.75.39	TCP		54 51111 → 443 [ACK] Seq=1371 Ack=42508 Win=131584…	0
58	11:16:49.524874	192.168.0.46	172.67.75.39	TCP		54 51111 → 443 [ACK] Seq=1371 Ack=48840 Win=131584…	0
59	11:16:49.561452	192.168.0.46	172.67.75.39	QUIC		1392 Initial, DCID=922cbb57ff4683e3, PKN: 1, CRYPTO,…	
65	11:16:49.606789	192.168.0.46	172.67.75.39	QUIC		95 Handshake, DCID=010727f5fc9976b540072bf7c299466…	
66	11:16:49.608420	192.168.0.46	172.67.75.39	QUIC		217 Protected Payload (KP0), DCID=010727f5fc9976b54…	
67	11:16:49.608912	192.168.0.46	172.67.75.39	QUIC		686 Protected Payload (KP0), DCID=010727f5fc9976b54…	
68	11:16:49.609176	192.168.0.46	172.67.75.39	QUIC		655 Protected Payload (KP0), DCID=010727f5fc9976b54…	
69	11:16:49.609514	192.168.0.46	172.67.75.39	QUIC		690 Protected Payload (KP0), DCID=010727f5fc9976b54…	
70	11:16:49.609779	192.168.0.46	172.67.75.39	QUIC		687 Protected Payload (KP0), DCID=010727f5fc9976b54…	
71	11:16:49.610023	192.168.0.46	172.67.75.39	QUIC		689 Protected Payload (KP0), DCID=010727f5fc9976b54…	

Figura 6. Columna agregada

6. Elimine u oculte la columna Longitud



No.	Time	Source	Destination	Protocol	Info	Long. TCP Segment
1	11:16:47.126585	192.168.0.46	172.67.75.39	TCP	51111 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=146…	0
3	11:16:47.177959	192.168.0.46	172.67.75.39	TCP	51111 → 443 [ACK] Seq=1 Ack=1 Win=131584 Len=0	0
4	11:16:47.178243	192.168.0.46	172.67.75.39	TLSv1.3	Client Hello (SNI=www.wireshark.org)	517
8	11:16:47.267999	192.168.0.46	172.67.75.39	TCP	51111 → 443 [ACK] Seq=518 Ack=1809 Win=131584 L…	0
9	11:16:49.048600	192.168.0.46	172.67.75.39	TLSv1.3	Change Cipher Spec, Application Data	64
10	11:16:49.048918	192.168.0.46	172.67.75.39	TLSv1.3	Application Data	92
11	11:16:49.049309	192.168.0.46	172.67.75.39	TLSv1.3	Application Data	666
14	11:16:49.074157	192.168.0.46	172.67.75.39	TLSv1.3	Application Data	31
19	11:16:49.123997	192.168.0.46	172.67.75.39	TCP	51111 → 443 [ACK] Seq=1371 Ack=2368 Win=130816 …	0
27	11:16:49.389546	192.168.0.46	172.67.75.39	TCP	51111 → 443 [ACK] Seq=1371 Ack=10871 Win=131584…	0
47	11:16:49.465037	192.168.0.46	172.67.75.39	TCP	51111 → 443 [ACK] Seq=1371 Ack=37921 Win=131584…	0
52	11:16:49.465309	192.168.0.46	172.67.75.39	TCP	51111 → 443 [ACK] Seq=1371 Ack=42508 Win=131584…	0
58	11:16:49.524874	192.168.0.46	172.67.75.39	TCP	51111 → 443 [ACK] Seq=1371 Ack=48840 Win=131584…	0
59	11:16:49.561452	192.168.0.46	172.67.75.39	QUIC	Initial, DCID=922cbb57ff4683e3, PKN: 1, CRYPTO,…	
65	11:16:49.606789	192.168.0.46	172.67.75.39	QUIC	Handshake, DCID=010727f5fc9976b540072bf7c299466…	
66	11:16:49.608420	192.168.0.46	172.67.75.39	QUIC	Protected Payload (KP0), DCID=010727f5fc9976b54…	
67	11:16:49.608912	192.168.0.46	172.67.75.39	QUIC	Protected Payload (KP0), DCID=010727f5fc9976b54…	
68	11:16:49.609176	192.168.0.46	172.67.75.39	QUIC	Protected Payload (KP0), DCID=010727f5fc9976b54…	
69	11:16:49.609514	192.168.0.46	172.67.75.39	QUIC	Protected Payload (KP0), DCID=010727f5fc9976b54…	
70	11:16:49.609779	192.168.0.46	172.67.75.39	QUIC	Protected Payload (KP0), DCID=010727f5fc9976b54…	
71	11:16:49.610023	192.168.0.46	172.67.75.39	QUIC	Protected Payload (KP0), DCID=010727f5fc9976b54…	

Figura 7. Columna oculta

7. Aplique un esquema de paneles que sea de su preferencia (que no sea el esquema por defecto)

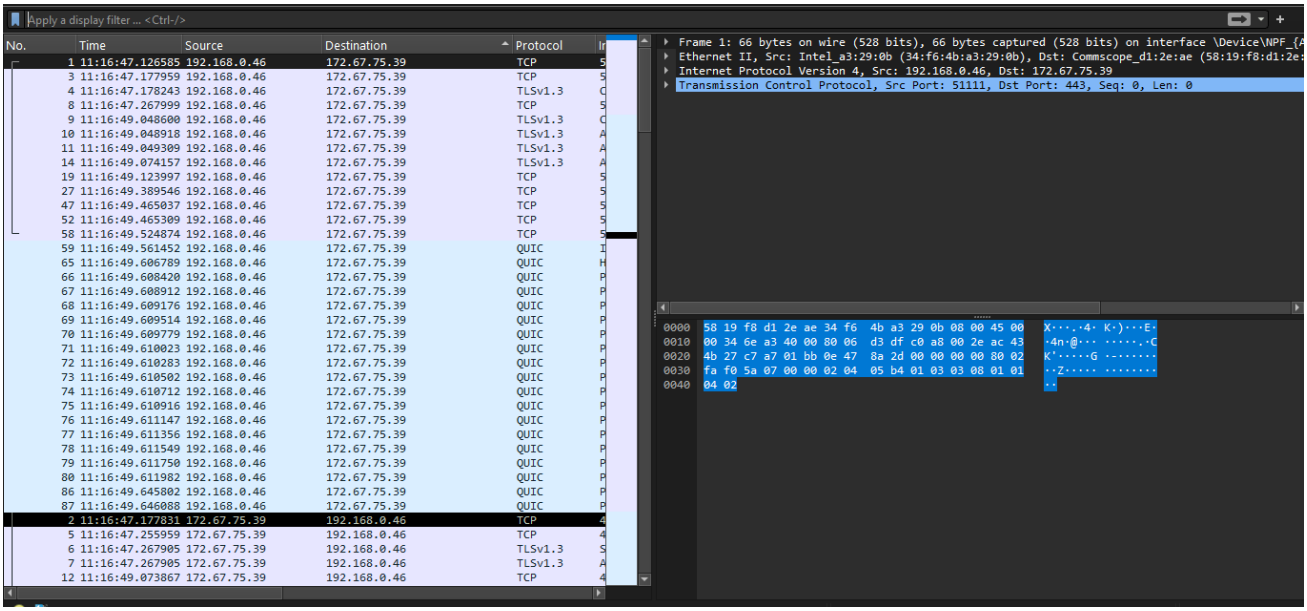


Figura 8. Nuevo Layout

8. Aplique una regla de color para los paquetes TCP cuyas banderas SYN sean iguales a 1, y coloque el color de su preferencia (View -> Coloring Rules)

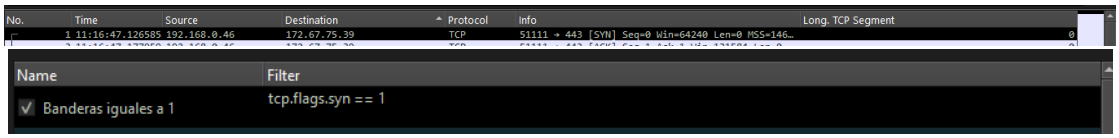


Figura 9. Nueva Regla

9. Cree un botón que aplique un filtro para paquetes TCP con la bandera SYN igual a 1.

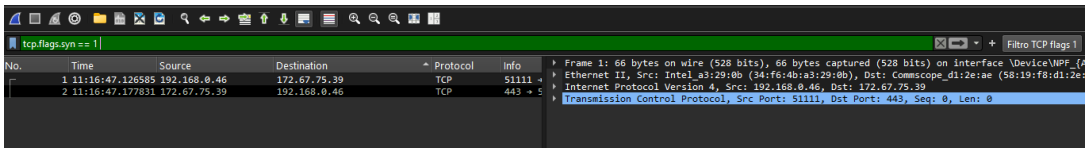


Figura 10. Botón con filtro aplicado

10. Oculte las interfaces virtuales (en caso aplique)

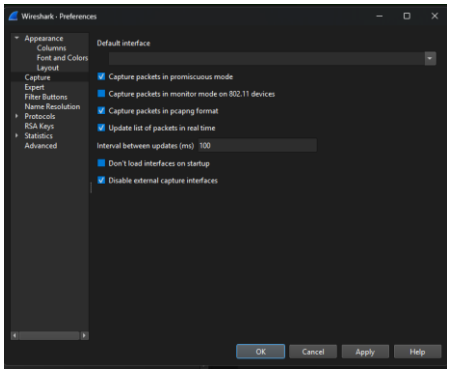


Figura 11. Interfaces

Se debe realizar tomas de pantalla que muestren el entorno final personalizado, el nombre del perfil y el uso de las reglas de color y botón del filtro, así como la lista simplificada de las interfaces de captura.

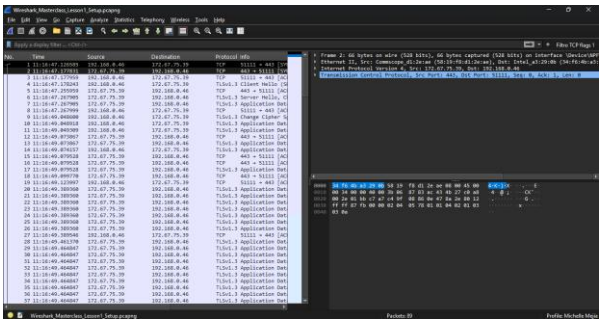


Figura 11. Resultado Final

1.2 Segunda parte: configuración de la captura de paquetes

En la segunda parte, se realizará una captura de paquetes con un ring buffer.

1. Abra una terminal y ejecute el comando ifconfig/ipconfig (dependiendo de su SO). Detalle y explique lo observado, investigue (i.e.: 'man ifconfig', documentación) de ser necesario. ¿Cuál es su interfaz de red?

   Mi interfaz principal es la "Wireless LAN adapter Wi-Fi", que es la que uso para conectarme a la red inalámbrica. También noté que hay otras interfaces, como "Conexión de área local 1" y "Conexión de área local 10", que están marcadas como "Disconnected", lo que significa que no están activas en ese momento, probablemente sean interfaces virtuales o adaptadores no utilizados.

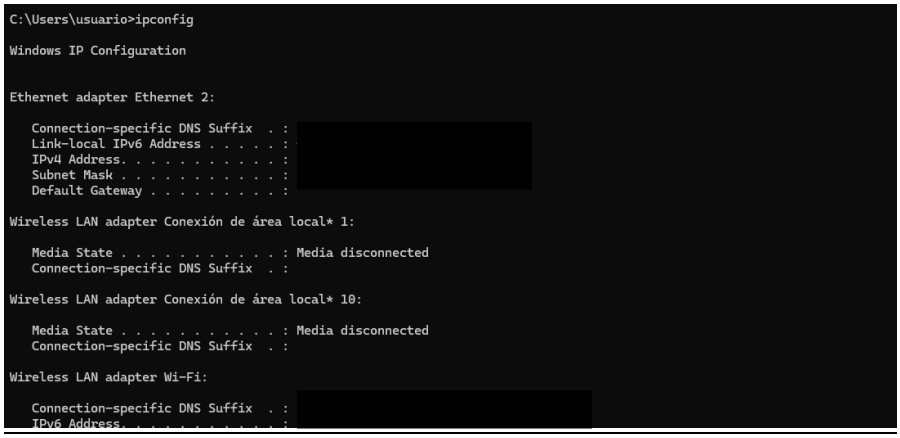
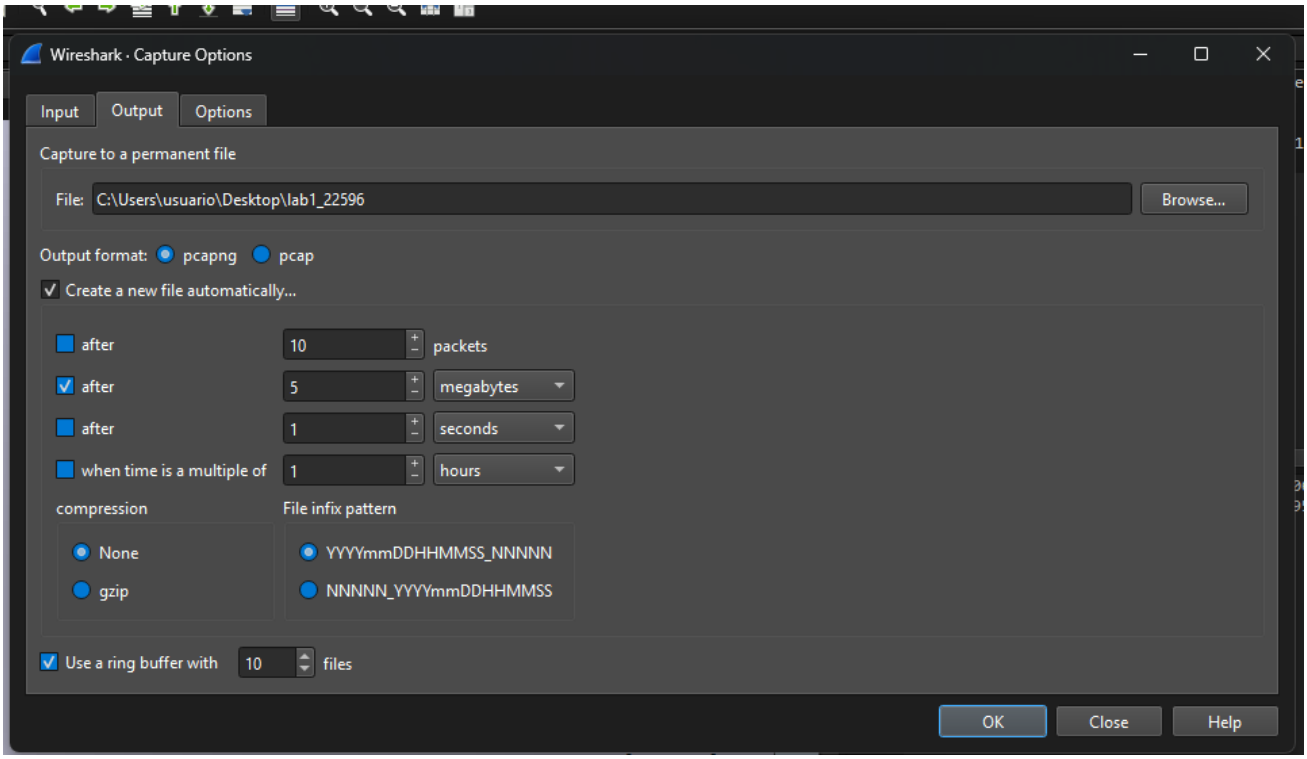


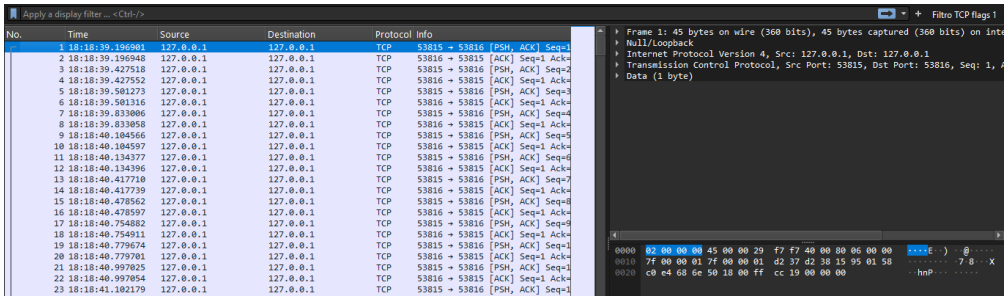
Figura 12. Resultado ipconfig

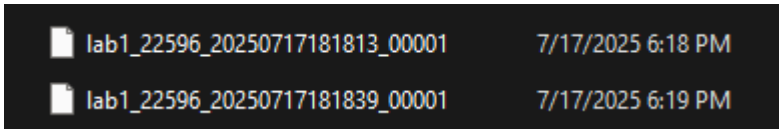
2. Luego, retornando a Wireshark, desactive las interfaces virtuales o que no aplique.

3. Realice una captura de paquetes con la interfaz de Ethernet o WiFi con una configuración de ring buffer, con un tamaño de 5 MB por archivo y un número máximo de 10 archivos (Capture -> Options -> Output) Genere tráfico para que los archivos se creen. Defina el nombre de los archivos de la siguiente forma: lab1_carnet.pgcap



Se debe realizar tomas de pantalla de la configuración o comandos para la creación del ring buffer, así como los archivos generados.

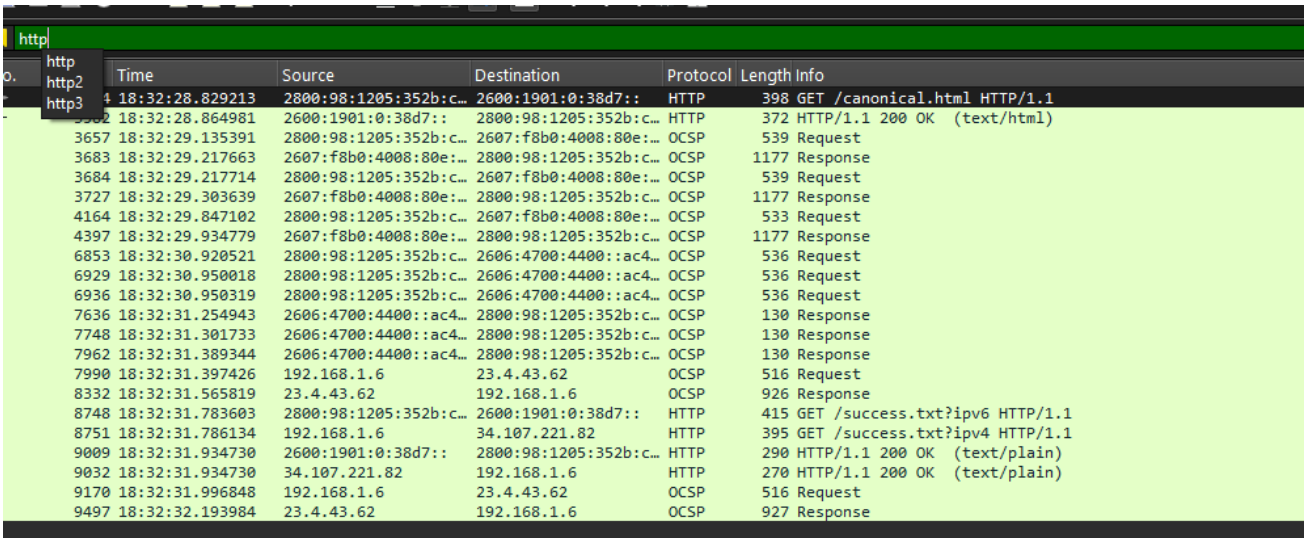




1.3 Tercera parte: análisis de paquetes

En la tercera parte se analizará el protocolo HTTP. Debe realizar tomas de pantalla que validen sus respuestas.

1. Abra su navegador, inicie una captura de paquetes en Wireshark (sin filtro) y acceda a la siguiente direccion (Si por alguna razón debe repetir el paso, borre su caché o utiliza el modo incógnito de su navegador): http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html
2. Detenga la captura de paquetes (si desea realizar una nueva captura de la página deberá borrar el caché de su navegador, de lo contrario no se realizará la captura del protocolo HTTP).



3. Responda las siguientes preguntas:
    a. ¿Qué versión de HTTP está ejecutando su navegador?
    HTTP/1.1 (en la solicitud GET)

    b. ¿Qué version de HTTP está ejecutando el servidor?
    El servidor está ejecutando **HTTP/1.1**, como lo indica el paquete de respuesta: HTTP/1.1 200 OK.

    c. ¿Qué lenguajes (si aplica) indica el navegador que acepta a el servidor?
    en-US

    d. ¿Cuántos bytes de contenido fueron devueltos por el servidor?
    bytes en text/html y 290 bytes en text/plain.

    e. En el caso que haya un problema de rendimiento mientras se descarga la página, ¿en que dispositivos de la red convendría “escuchar” los paquetes? ¿Es conveniente instalar Wireshark en el servidor? Justifique.

    Si hay un problema de rendimiento al descargar una página, creo que lo más conveniente es escuchar los paquetes en el cliente, ya que es desde ahí donde se experimenta la lentitud. Si la red o el dispositivo del usuario tiene problemas, como paquetes perdidos o alta latencia, se puede identificar fácilmente desde el cliente. También sería útil escuchar en el servidor, para ver si el problema está relacionado con la capacidad del servidor para procesar solicitudes o enviar respuestas rápidamente. Si el servidor está sobrecargado o tiene cuellos de botella, lo sabríamos analizando el tráfico directamente ahí. En cuanto a instalar Wireshark en el servidor, no lo considero recomendable para un servidor de producción, porque puede generar una sobrecarga en el servidor y afectar su rendimiento. En lugar de eso, creo que sería mejor usar herramientas más ligeras y específicas para monitorear el tráfico sin afectar el rendimiento, como netstat o sistemas de monitoreo como Zabbix.

Discusión de Resultados

La actividad consistió en capturar paquetes de red usando Wireshark , y me permitió obtener una visión más clara de cómo los dispositivos de la red se comunican. Durante la actividad, aprendí a identificar diferentes tipos de tráfico de red, como consultas DNS y solicitudes/respuestas HTTP. Lo más interesante fue ver cómo la red maneja las solicitudes entre el cliente y el servidor, y cómo diferentes factores (como el tipo de protocolo o la latencia de la red) afectan el rendimiento. En particular, descubrí que, en muchos casos, los problemas de rendimiento no necesariamente provienen del servidor, sino que pueden ser causados por la red o el dispositivo cliente.

En cuanto a la parte práctica, me di cuenta de que Wireshark es una herramienta poderosa para analizar y diagnosticar problemas de red, pero también puede ser abrumadora si no se sabe cómo filtrar y analizar correctamente los paquetes. Por ejemplo, pude identificar la versión HTTP utilizada por el navegador y el servidor, así como obtener detalles sobre los bytes devueltos por el servidor. Sin embargo, hubo algunos desafíos al tratar de capturar y analizar paquetes relacionados con el rendimiento en tiempo real, especialmente cuando no había tráfico de red suficiente.

Comentarios:

Una de las cosas que más me llamó la atención fue la flexibilidad de Wireshark para capturar y analizar tráfico. Me sorprendió lo fácil que es visualizar los paquetes HTTP y ver cómo se comporta la comunicación entre un cliente y un servidor. Sin embargo, al principio me costó un poco acostumbrarme a cómo filtrar correctamente los paquetes, ya que Wireshark captura una gran cantidad de datos, y era fácil perderse entre ellos. Creo que con más práctica y tiempo, esta herramienta me será muy útil en futuras investigaciones o proyectos relacionados con redes.

Conclusiones:

1. Wireshark es una herramienta esencial para el análisis de redes, ya que permite observar en detalle el tráfico entre dispositivos y servidores, ayudando a identificar posibles cuellos de botella o problemas de rendimiento.

2. La captura de paquetes HTTP y DNS me permitió entender cómo las solicitudes de los usuarios son gestionadas por los servidores y cómo se manejan las respuestas, como los códigos de estado HTTP .

3. Instalar Wireshark en el servidor puede ser útil en algunos casos, pero no es recomendable en un entorno de producción debido a la sobrecarga que puede generar en el rendimiento del servidor.

4. La actividad también me enseñó la importancia de filtrar los paquetes adecuadamente y de comprender cómo se estructuran los protocolos de red para hacer análisis más eficientes.

Referencias

Wireshark Foundation. (s.f.). *Documentación de Wireshark*. Recuperado de https://www.wireshark.org/docs/

Internet Engineering Task Force (IETF). (1999). *RFC 2616: Hypertext Transfer Protocol (HTTP/1.1)*. Recuperado de https://datatracker.ietf.org/doc/html/rfc2616

Internet Engineering Task Force (IETF). (2015). *RFC 7230: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. Recuperado de https://datatracker.ietf.org/doc/html/rfc7230