

Universidad del Valle de Guatemala  
Facultad de Ingeniería  
Redes  
Profesor: Jorge Yass

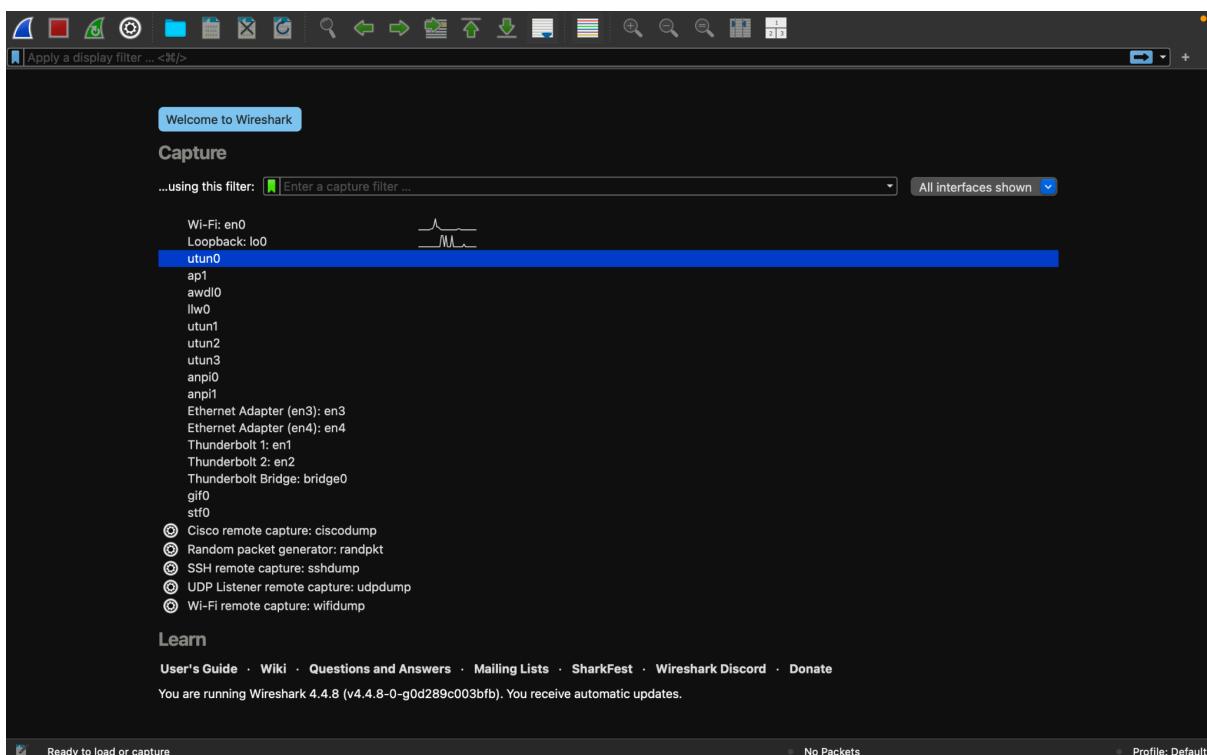


## **“Lab parte 2 Redes”**

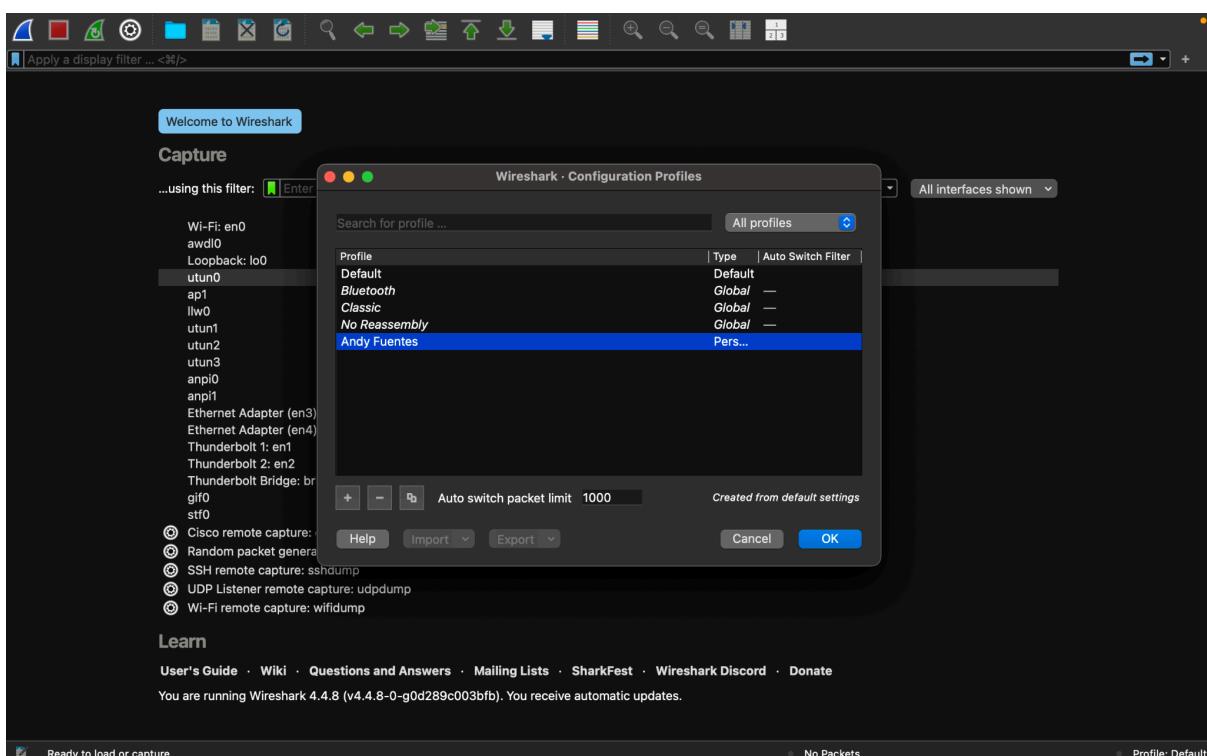
Andy Fuentes 22944

17 de julio del 2025, Guatemala de la Asunción

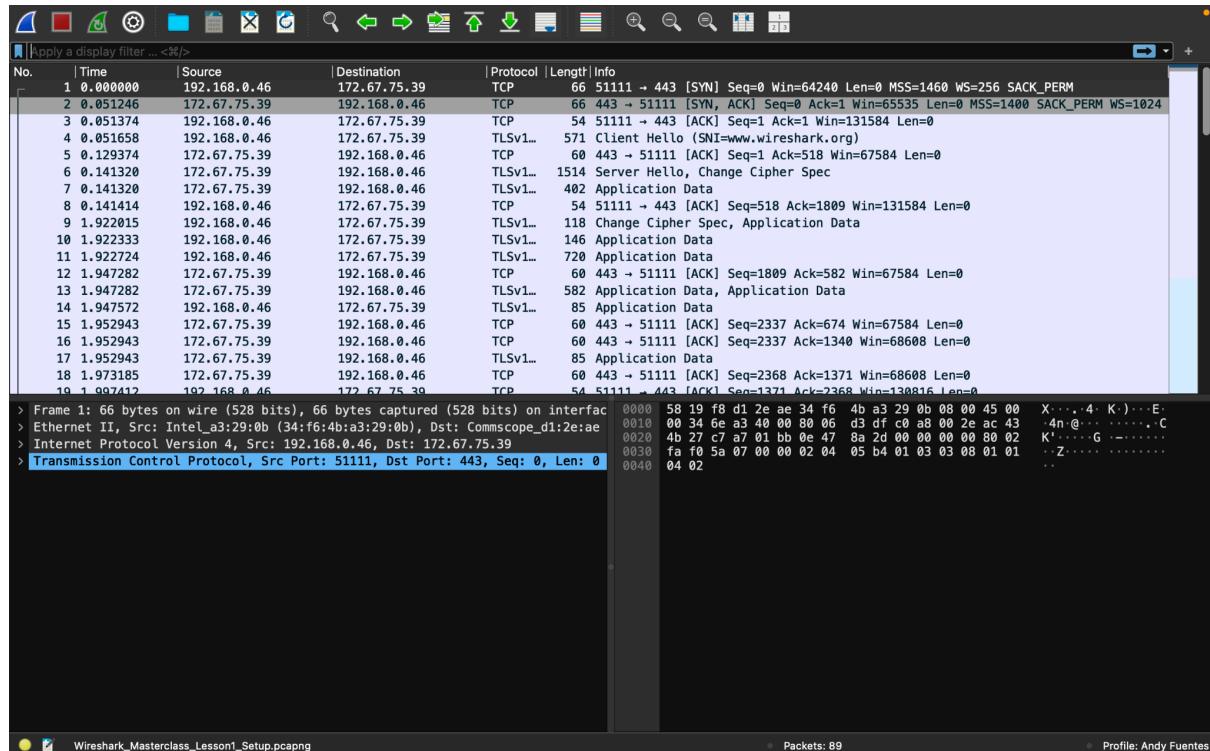
## Instalación wireshark



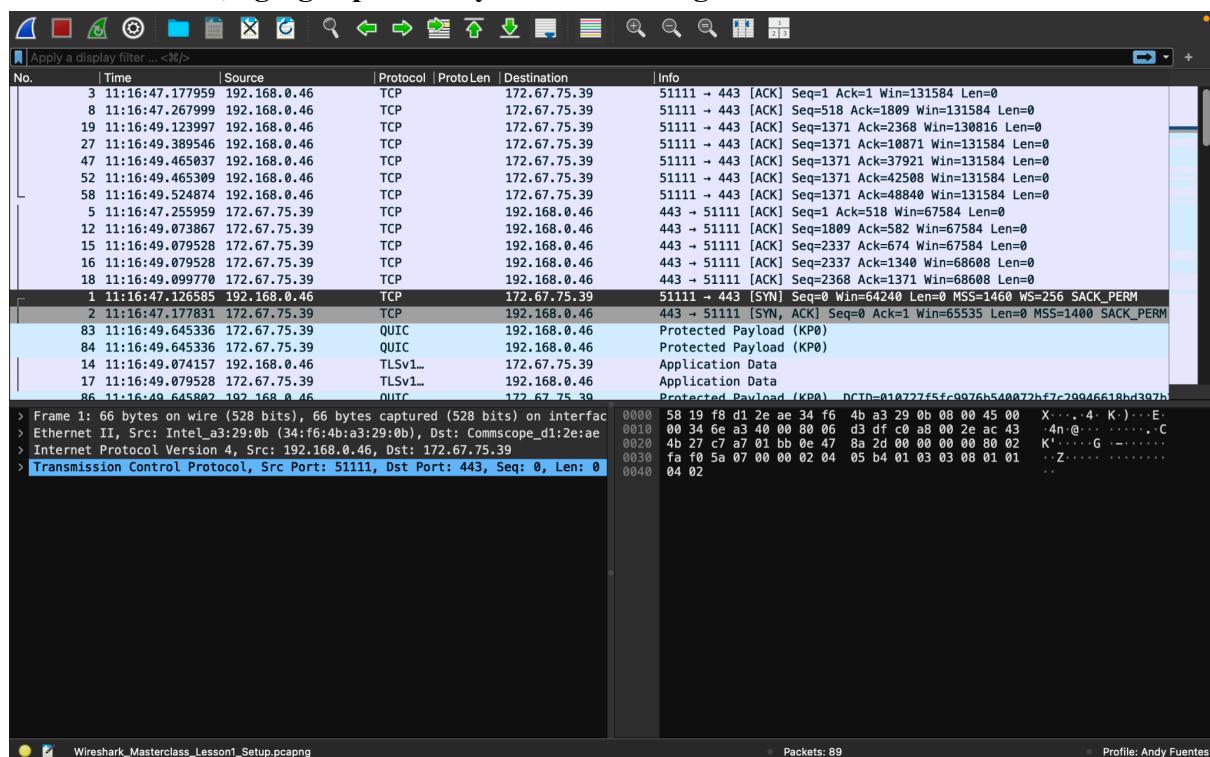
## Perfil Activo



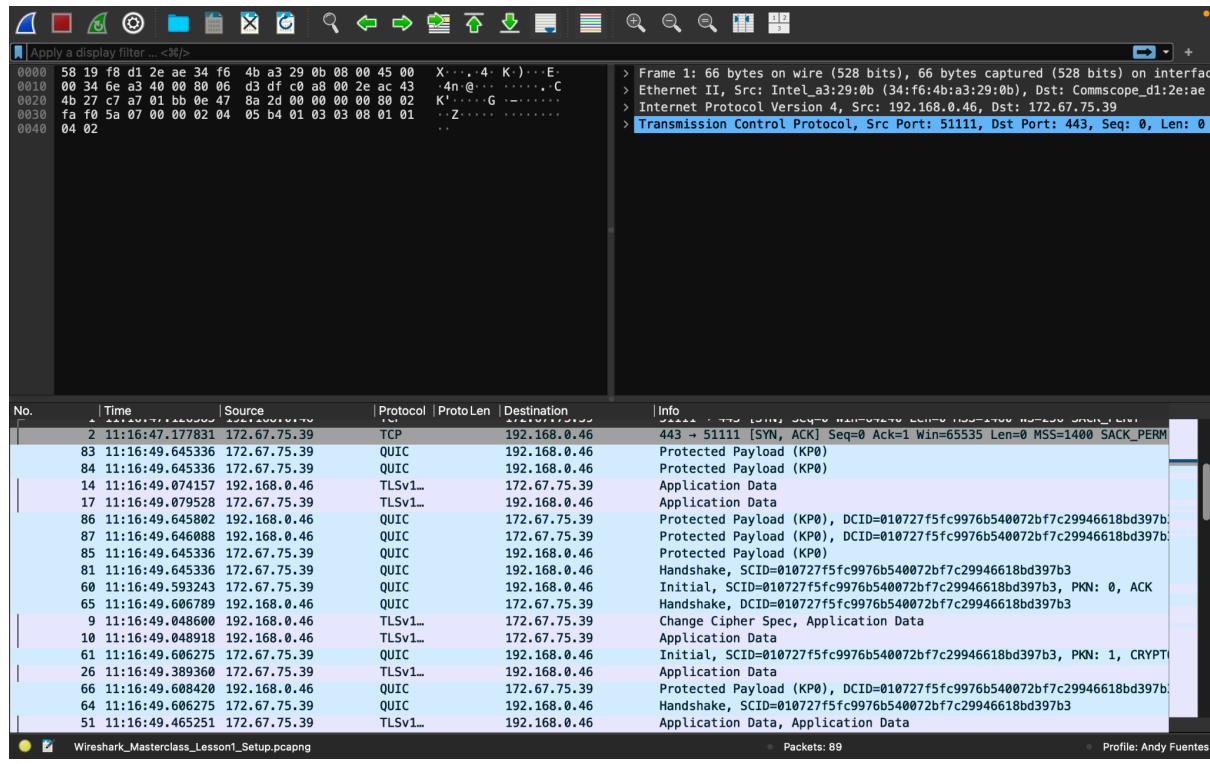
## Imagen del wireshark descargada:



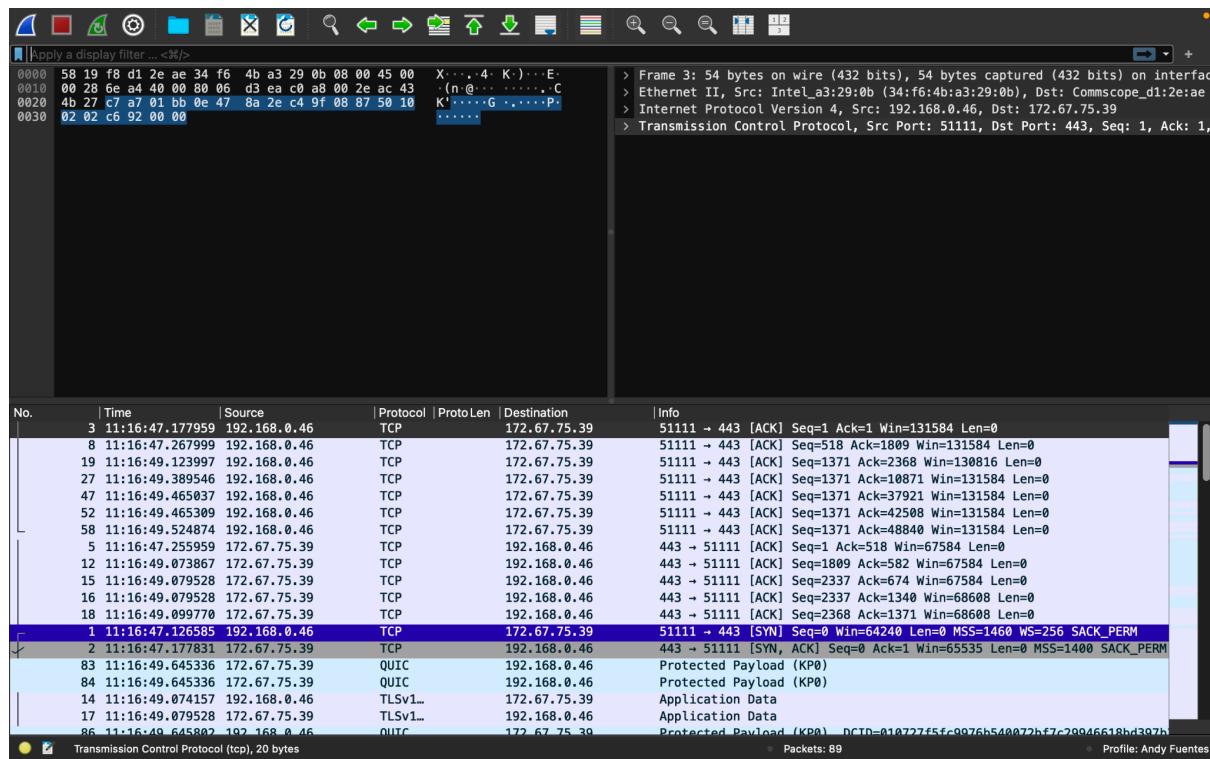
## Cambio de time, agregar protolen y desmarcar lenght

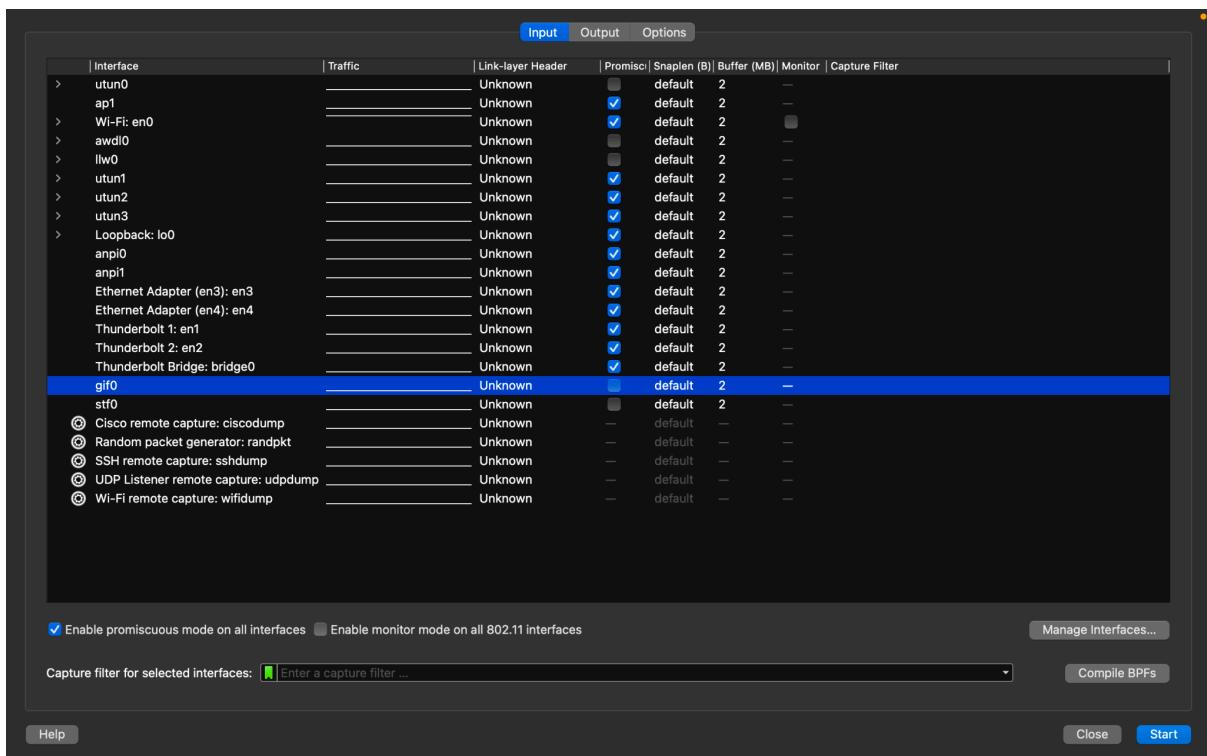


## Cambio panel



## Botón de filtro SYN





## Parte 1.2

No.	Time	Source	Protocol	ProtoLen	Destination	Info
23	23:01:07.513507	192.168.0.9	SSDP	239.255.255.250		57621 ~ 57621 Len=44
24	23:01:07.508827	192.168.0.9	SSDP	239.255.255.250		M-SEARCH * HTTP/1.1
25	23:01:08.512346	192.168.0.9	SSDP	239.255.255.250		M-SEARCH * HTTP/1.1
26	23:01:09.513507	192.168.0.9	SSDP	239.255.255.250		M-SEARCH * HTTP/1.1
27	23:01:10.448808	192.168.0.9	UDP	192.168.0.255		57621 ~ 57621 Len=44
28	23:01:10.514550	192.168.0.9	SSDP	239.255.255.250		M-SEARCH * HTTP/1.1
29	23:01:10.449733	192.168.0.9	UDP	192.168.0.255		57621 ~ 57621 Len=44
30	23:02:04.485619	192.168.0.9	MDNS	224.0.0.251		Standard query 0x0000 PTR _spotify-connect._tcp.local. "QM" question
31	23:02:04.486855	fe80::f1d4e7:72b::	MDNS	ff02::fb		Standard query 0x0000 PTR _spotify-connect._tcp.local. "QM" question
32	23:02:04.486949	fe80::ccec:b4ff:fe::	MDNS	ff02::fb		Standard query 0x0000 PTR _spotify-connect._tcp.local. "QM" question
33	23:02:04.487049	fe80::ccec:b4ff:fe::	MDNS	ff02::fb		Standard query 0x0000 PTR _spotify-connect._tcp.local. "QM" question
34	23:02:04.662165	192.168.0.9	SSDP	239.255.255.250		M-SEARCH * HTTP/1.1
35	23:02:10.450497	192.168.0.9	UDP	192.168.0.255		57621 ~ 57621 Len=44
36	23:02:40.451487	192.168.0.9	UDP	192.168.0.255		57621 ~ 57621 Len=44
37	23:03:07.503256	192.168.0.9	SSDP	239.255.255.250		M-SEARCH * HTTP/1.1
38	23:03:08.504333	192.168.0.9	SSDP	239.255.255.250		M-SEARCH * HTTP/1.1
39	23:03:09.505385	192.168.0.9	SSDP	239.255.255.250		M-SEARCH * HTTP/1.1
40	23:03:10.452384	192.168.0.9	UDP	192.168.0.255		57621 ~ 57621 Len=44
41	23:03:10.506568	192.168.0.9	SSDP	239.255.255.250		M-SEARCH * HTTP/1.1

Loopback: lo0: <live capture in progress>

Packets: 41 Profile: Andy Fuentes

	Lab1_A00456_20250717225950_00001.pcapng	9 KB	Pcap...Capture	hoy, 10:59 p. m.
	Lab1_A00456_20250717225346_00001.pcapng	256 bytes	Pcap...Capture	hoy, 10:53 p. m.
	Lab1_A00456_20250717225716_00001.pcapng	256 bytes	Pcap...Capture	hoy, 10:57 p. m.
	Lab1_A00456_20250717225749_00001.pcapng	248 bytes	Pcap...Capture	hoy, 10:57 p. m.
	Lab1_A00456_20250717224739_00001.pcapng	236 bytes	Pcap...Capture	hoy, 10:47 p. m.

Apply a display filter: <%>

No.	Time	Source	Protocol	Proto Len	Destination	Info
1	23:14:04.478070	192.168.0.9	MDNS	224.0.0.251		Standard query 0x0000 PTR _spotify-connect._tcp.local, "QM" question
2	23:14:04.478291	fe80::6f:d4e7:72b::	MDNS	ff02::fb		Standard query 0x0000 PTR _spotify-connect._tcp.local, "QM" question
3	23:14:04.478459	fe80::ccec:b4ff:fe::	MDNS	ff02::fb		Standard query 0x0000 PTR _spotify-connect._tcp.local, "QM" question
4	23:14:04.478591	fe80::ccec:b4ff:fe::	MDNS	ff02::fb		Standard query 0x0000 PTR _spotify-connect._tcp.local, "QM" question
5	23:14:04.664761	192.168.0.9	SSDP	239.255.255.250		M-SEARCH * HTTP/1.1
6	23:14:10.485114	192.168.0.9	UDP	192.168.0.255		57621 → 57621 Len=44

### 1.3

Apply a display filter: <%>

No.	Time	Source	Protocol	Proto Len	Destination	Info
1	23:22:40.703339	192.168.0.9	UDP	192.168.0.255		57621 → 57621 Len=44
2	23:23:07.542477	192.168.0.9	SSDP	239.255.255.250		M-SEARCH * HTTP/1.1
3	23:23:08.544238	192.168.0.9	SSDP	239.255.255.250		M-SEARCH * HTTP/1.1
4	23:23:09.544930	192.168.0.9	SSDP	239.255.255.250		M-SEARCH * HTTP/1.1
5	23:23:10.545493	192.168.0.9	SSDP	239.255.255.250		M-SEARCH * HTTP/1.1
6	23:23:10.746605	192.168.0.9	UDP	192.168.0.255		57621 → 57621 Len=44

### Análisis:

En la captura realizada sobre la interfaz de loopback (lo0) se observa principalmente tráfico de descubrimiento de servicios locales: por un lado, paquetes UDP broadcast dirigidos a la dirección 192.168.0.255 (puerto 57621 → 57621, longitud de 44 bytes), que corresponden a mensajes de anuncio o consulta de un servicio propio de la máquina; y por otro, varias solicitudes SSDP “M-SEARCH \* HTTP/1.1” enviadas a la dirección multicast 239.255.255.250, empleadas para detectar dispositivos y servicios UPnP en la red local. Al tratarse de comunicaciones de red interna, no aparecen aquí peticiones HTTP estándar hacia servidores remotos—estas solo se capturan al monitorizar la interfaz física (en0). La presencia de estos protocolos de descubrimiento indica que la máquina está intentando localizar otros nodos y servicios en su subred antes de establecer conexiones basadas en TCP o HTTP.

### **a. Versión HTTP del navegador**

Al inspeccionar la primera línea de la petición registrada, encontramos un encabezado que inicia con

GET /wireshark-labs/INTRO-wireshark-file1.html HTTP/1.1

Esto confirma que el navegador está utilizando la versión HTTP/1.1 para comunicarse con el servidor.

### **b. Versión HTTP del servidor**

En la respuesta correspondiente al GET anterior, el servidor devuelve una línea de estado que comienza por

HTTP/1.1 200 OK

De modo que, al igual que el cliente, el servidor también está ejecutando HTTP/1.1.

### **c. Lenguajes que acepta el navegador**

Dentro de los encabezados de la petición, aparece la línea

Accept-Language: en-US,en;q=0.5

lo cual indica que el navegador prefiere contenido en inglés de Estados Unidos y acepta también otros dialectos de inglés con un factor de calidad (“q”) de 0.5.

### **d. Bytes de contenido devueltos por el servidor**

En la sección de encabezados de la respuesta HTTP encontramos

Content-Length: 4832

Este valor nos dice que el servidor ha enviado 4832 bytes de contenido en el cuerpo de la respuesta.

### **e. Dónde “escuchar” los paquetes y si conviene instalar Wireshark en el servidor**

Para diagnosticar problemas de rendimiento (latencia, pérdidas o retransmisiones) es más efectivo capturar el tráfico en puntos clave de la red, como el enlace cliente→switch y el enlace switch→servidor, utilizando un puerto mirror (SPAN) o un dispositivo de sniffing dedicado. Instalar Wireshark directamente en el servidor de producción no es recomendable, ya que el propio proceso de captura consume recursos de CPU y memoria, lo que podría alterar los tiempos de respuesta que se pretenden medir y añadir sesgo a los resultados.

## **Discusión sobre la actividad, experiencia y hallazgos**

La primera parte del laboratorio (personalización del entorno) permitió familiarizarme con las opciones avanzadas de Wireshark: creación de perfiles, ajuste del formato de tiempo, columnas personalizadas (“Proto Len”), esquemas de paneles, reglas de color y botones de filtro. Gracias a ello pude optimizar la visibilidad del tráfico que me interesaba—en especial los paquetes TCP SYN.

En la segunda parte, la configuración de un ring-buffer para captura automática destacó la potencia de Wireshark para entornos de alto volumen: definir un archivo base, límite de 5 MB y rotación de 10 archivos garantiza no perder datos cuando se monitoriza por largos períodos. Aquí encontré un reto práctico al seleccionar la interfaz correcta: inicialmente capturaba en gif0 o lo0 y no veía nada de tráfico HTTP, lo que me obligó a investigar la lista de interfaces con ifconfig y usar la interfaz física (en0) o, alternativamente, el loopback para pruebas locales.

Finalmente, el análisis HTTP de la tercera parte consolidó el entendimiento de cómo inspeccionar protocolos de aplicación: localizar la petición GET, extraer versiones HTTP, encabezados Accept-Language, Content-Length y comprender el flujo request-response. Este ejercicio resaltó la importancia de capturar en el punto de la red por donde realmente circula el tráfico de interés.

## Comentarios

Interfaz virtual vs. física: macOS lista muchas interfaces túnel; distinguir cuál usar requirió consultar ifconfig.

Modo monitor vs. managed: en Wi-Fi es frecuente no ver tráfico IP en modo “managed” sin activar el modo monitor.

Usabilidad de Wireshark: la gran cantidad de opciones puede abrumar al principio, pero combinar perfiles, filtros y color-rules agiliza mucho el análisis.

## Conclusiones

Este laboratorio demostró que, con una configuración adecuada, Wireshark es una herramienta extremadamente flexible tanto para tareas puntuales (ver un GET HTTP) como para monitoreos continuos sin intervención (ring-buffer). Crear un perfil dedicado y personalizar la interfaz de usuario ayuda a centrarse en los datos más relevantes. Asimismo, aprender a seleccionar correctamente la interfaz de captura es esencial para no filtrar tráfico erróneo. Finalmente, el análisis de HTTP me permitió reforzar conceptos de protocolos de capa de aplicación y buenas prácticas para diagnósticos de rendimiento.

## **Referencias Utilizadas**

Apple Inc. (n.d.). ifconfig(8) manual page. Recuperado el 17 de julio de 2025, de <https://developer.apple.com/library/archive/documentation/Darwin/Reference/ManPages/man/8/ifconfig.8.html>

Internet Engineering Task Force. (2014). RFC 7230: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. Recuperado de <https://tools.ietf.org/html/rfc7230>

UPnP Forum. (2017). Simple Service Discovery Protocol (SSDP). Recuperado de <https://openconnectivity.org/specs/upnp/2017/SSDP.pdf>

Wireshark Foundation. (n.d.-a). Wireshark User's Guide. Recuperado el 17 de julio de 2025, de [https://www.wireshark.org/docs/wsug\\_html\\_chunked/](https://www.wireshark.org/docs/wsug_html_chunked/)

Wireshark Foundation. (n.d.-b). Ring Buffer Options. En Wireshark User's Guide. Recuperado el 17 de julio de 2025, de [https://www.wireshark.org/docs/wsug\\_html\\_chunked/ChCaptureRingBuffer.html](https://www.wireshark.org/docs/wsug_html_chunked/ChCaptureRingBuffer.html)