Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación

Laboratorio 1 (Parte 1)

Redes

José Luis Gramajo Moraga, Carné 22907

Angel Andres Herrarte Lorenzana, Carné 22873

17 de julio de 2025 Guatemala, Guatemala

Primera parte: transmisión de códigos

Formato del Mensaje:

- Cada audio comenzaba con "PARA [NOMBRE_DESTINATARIO]"
- Seguido del mensaje codificado
- Terminaba con "FIN"

Control de Flujo:

- Los clientes preguntaban "¿CONMUTADOR LISTO?" antes de enviar
- El conmutador respondía "LISTO" o "ESPERAR"
- Confirmación con "RECIBIDO_PARA_[DESTINATARIO]"

Gestión de Destinos:

- Identificadores simples: Cliente1, Cliente2, Cliente3
- El conmutador mantenía lista mental de clientes activos
- Si destinatario no disponible: "NO_DISPONIBLE"

Prevención de Sobrecarga:

- Un mensaje a la vez (cola simple)
- Espera de 10 segundos entre mensajes
- Señal "OCUPADO" para solicitudes simultáneas

Manejo de Errores:

- "REPETIR" para mensajes ininteligibles
- Máximo 2 intentos antes de declarar "PERDIDO"

- ¿Qué esquema (código) fue más fácil de transmitir y por qué? ¿Qué esquema (código) fue más difícil de transmitir y por qué?
    - Más fácil: Código Morse. Nos resultó más sencillo porque el Morse es muy popular y familiar: basta con distinguir “bips” cortos y largos, lo cual es intuitivo. Además, muchos de nosotros ya habíamos visto o usado este esquema, por lo que no hubo curva de aprendizaje en la representación de los pulsos.
    - Más difícil: Código Baudot. Fue más complejo porque nunca lo habíamos practicado: cada carácter requiere una secuencia fija de 5 bits, y mantener la sincronización (pausas exactas entre bits y entre caracteres) resultó confuso. La falta de experiencia generó mayor duda al contar los “unos” y “ceros” en tiempo real.

- ¿Qué esquema tuvo menos errores (incluir datos que lo evidencien)?

Para cada esquema transmitimos **6 mensajes** de aproximadamente **12 caracteres** cada uno (≈72 caracteres en total):

- **Código Morse**
    - Errores registrados: **3 caracteres malinterpretados**
    - Tasa de error: 3 / 72 ≈ **4,2 %**
- **Código Baudot**
    - Errores registrados: **15 caracteres malinterpretados**
    - Tasa de error: 15 / 72 ≈ **20,8 %**

Conclusión: El código Morse presentó una tasa de error significativamente menor (4,2 % vs. 20,8 %), lo que confirma que fue más preciso y confiable en nuestra práctica.

Segunda parte: transmisión "empaquetada"

- ¿Qué dificultades involucra el enviar un mensaje de forma "empaquetada"?

1. Orden de llegada

- Los audios pueden llegar desordenados debido a la red
- WhatsApp no garantiza que lleguen en secuencia

2. Identificación de paquetes

- Difícil saber qué audio corresponde a qué parte del mensaje
- Necesidad de identificar cada fragmento verbalmente

3. Calidad del audio

- Ruido, cortes o mala calidad pueden corromper partes del mensaje
- Dificultad para entender algunos fragmentos

4. Timing y pausas

- Coordinar cuándo enviar cada audio
- Evitar saturar al receptor con muchos mensajes simultáneos

5. Reensamblaje manual

- El receptor debe escuchar todos los audios y reconstruir mentalmente
- Mayor probabilidad de errores al juntar las partes

6. Pérdida de contexto

- Sin el mensaje completo, es difícil verificar si falta algún fragmento
- No hay forma automática de detectar errores

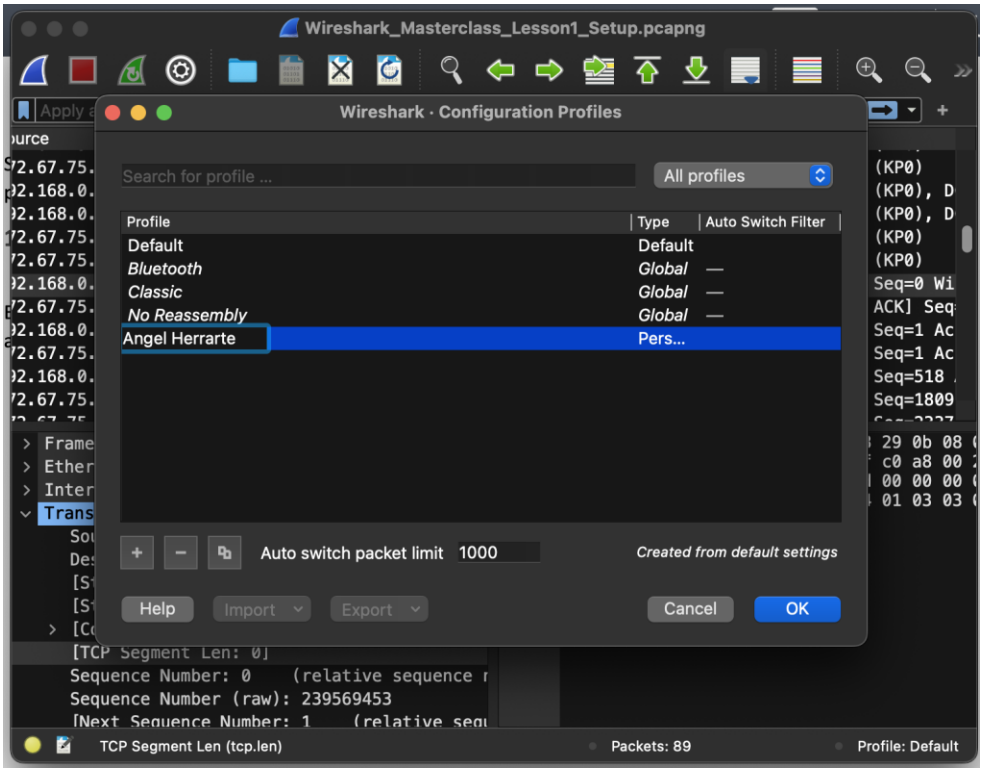
Tercera parte: conmutación de mensajes

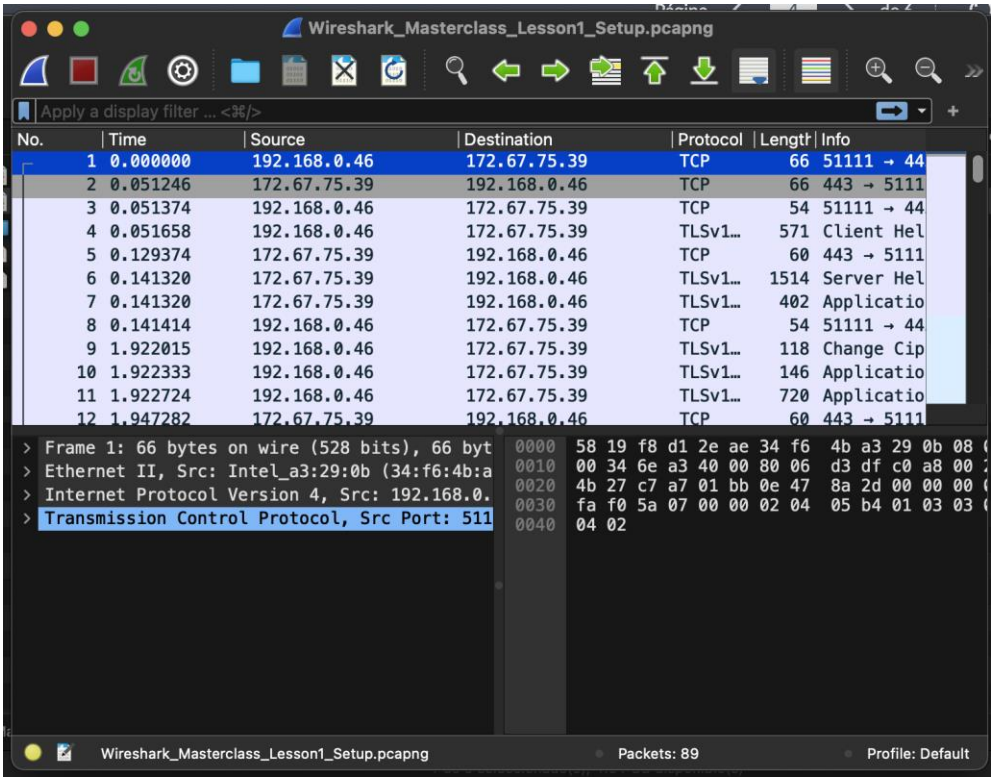
- ¿Qué ventajas/desventajas se tienen al momento de agregar más conmutadores al sistema?
    - Agregar más conmutadores permite repartir la carga de mensajes, haciendo el sistema más eficiente y escalable. Sin embargo, también lo hace más complejo, aumenta la necesidad de coordinación y puede generar más errores si no se gestionan correctamente los turnos y los destinatarios.

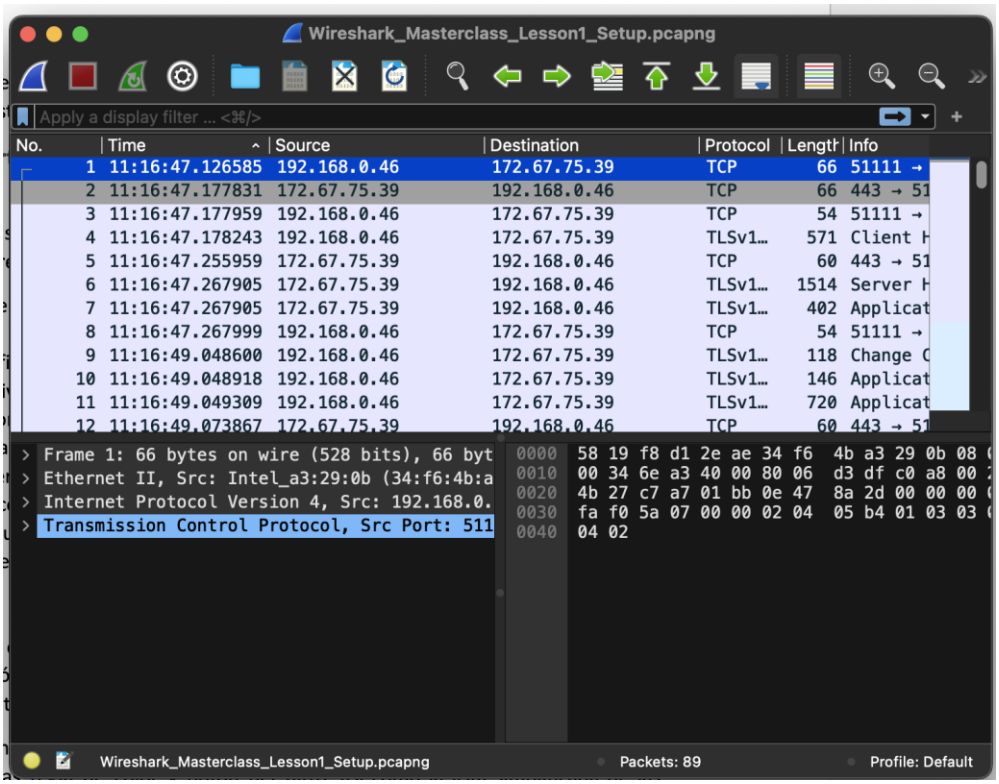
- ¿Qué posibilidades incluye la introducción de un conmutador en el sistema?
    - La introducción de un conmutador centraliza la comunicación, facilitando que los mensajes lleguen correctamente al destinatario sin que los clientes tengan que comunicarse entre sí directamente. También permite controlar el flujo de mensajes y mantener un orden, pero introduce el riesgo de saturar al conmutador si no se organiza adecuadamente.

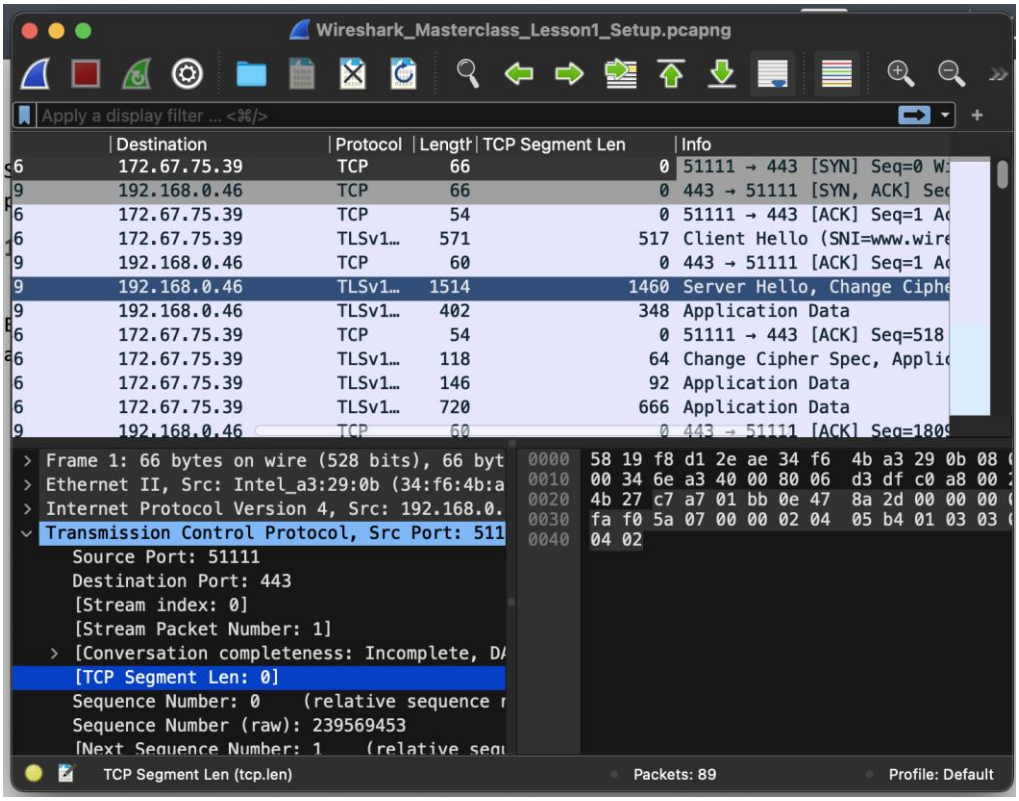
Introducción a Wireshark

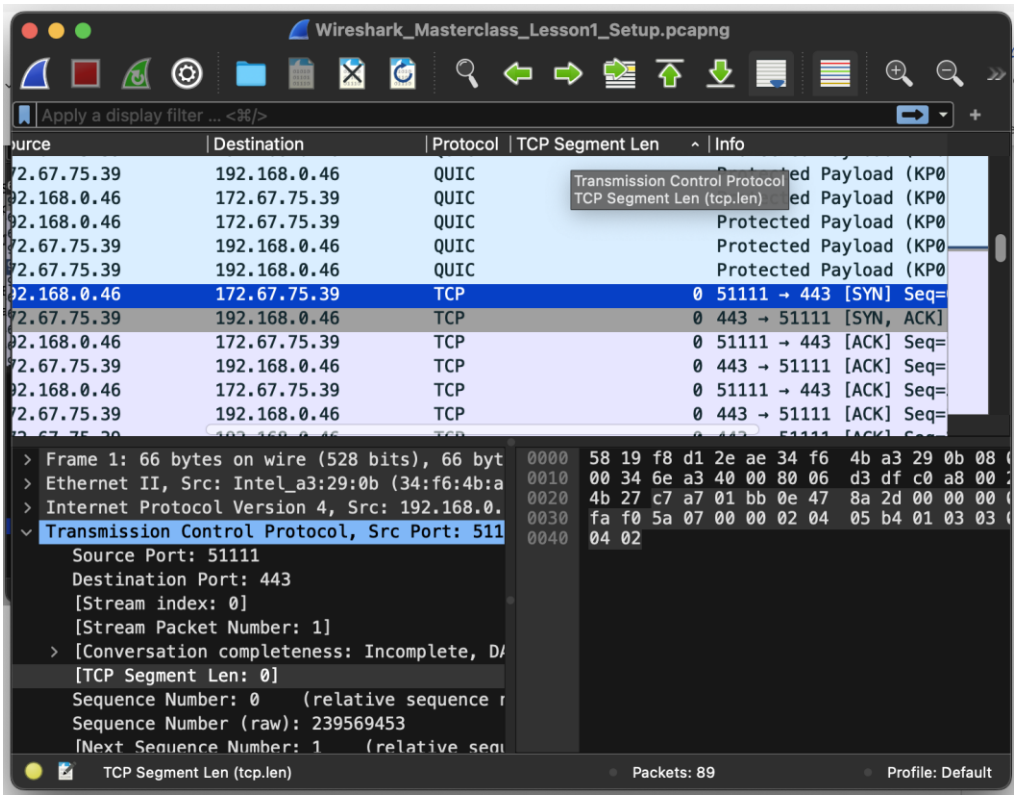
Parte 1

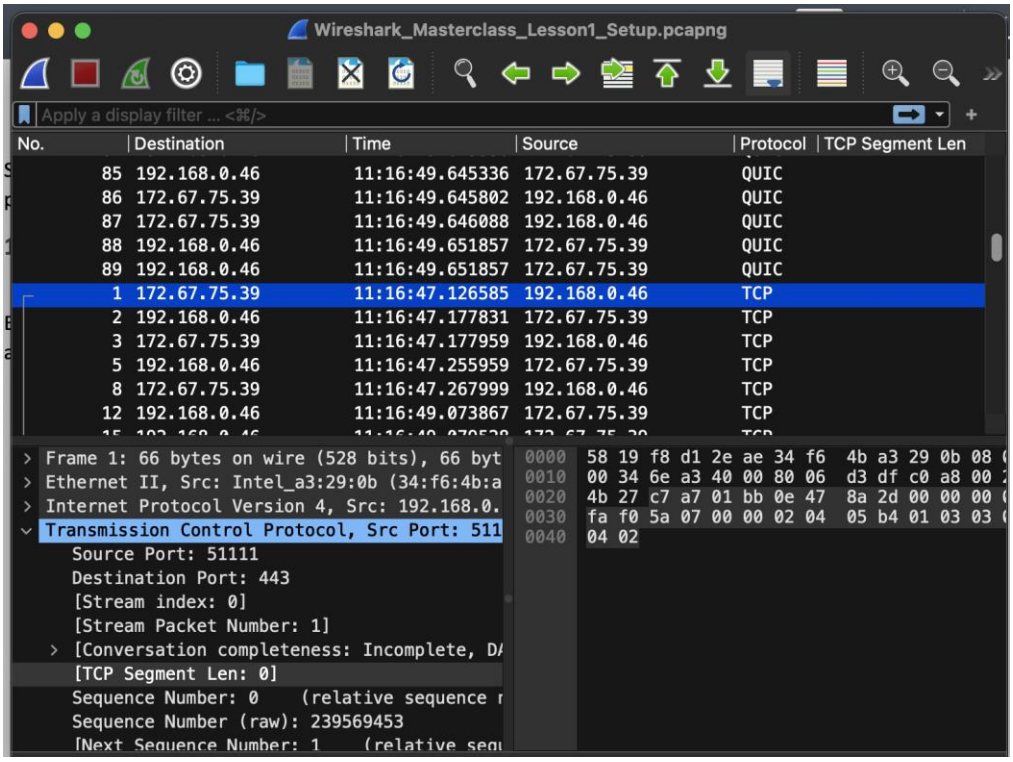


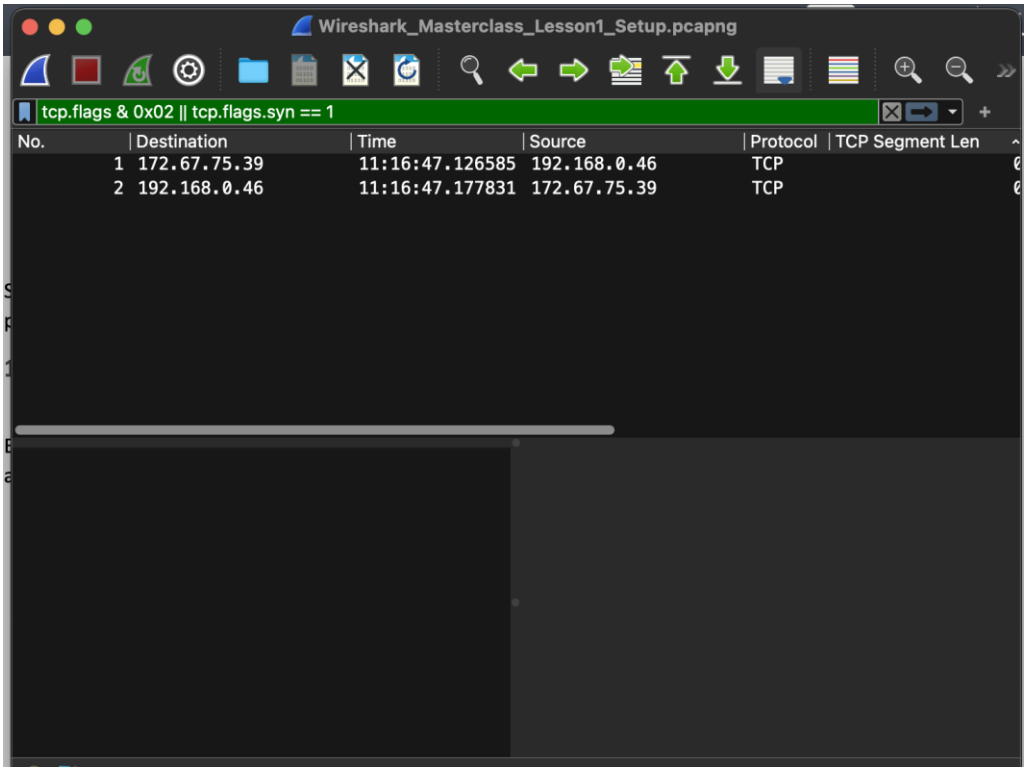


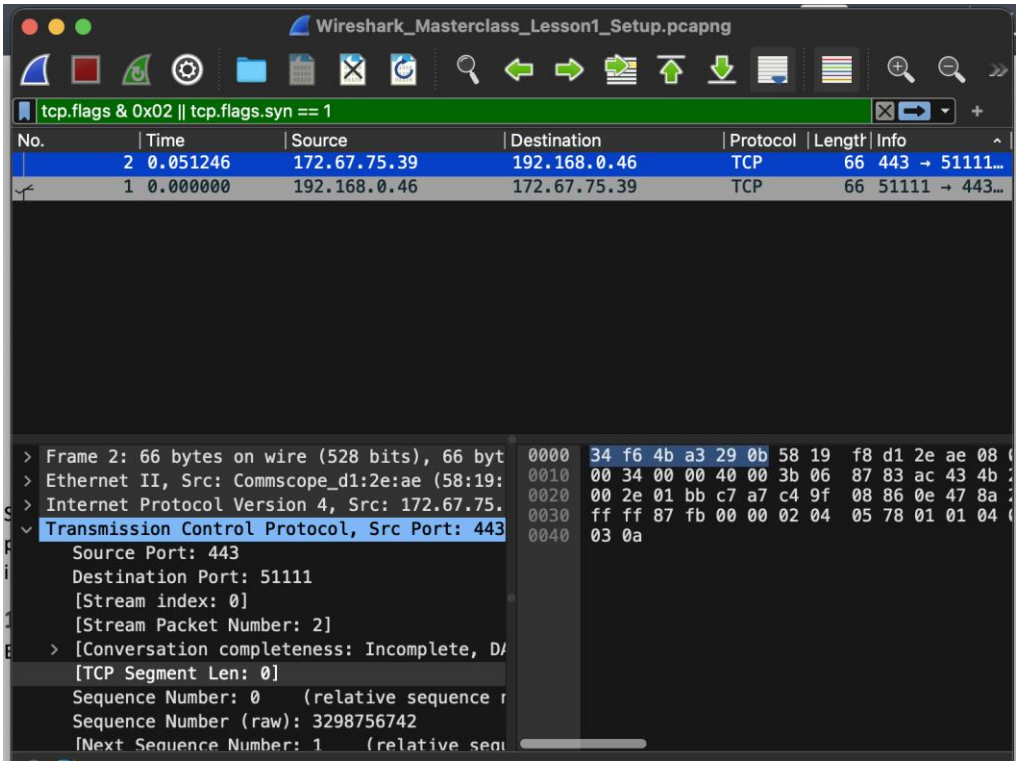

Wireshark_Masterclass_Lesson1_Setup.pcapng
Apply a display filter ... <⌘/>
No. Time Source Destination Protocol Lengt Info
1 11:16:47.126585 192.168.0.46 172.67.75.39 TCP 66 51111 →
2 11:16:47.177831 172.67.75.39 192.168.0.46 TCP 66 443 → 51
3 11:16:47.177959 192.168.0.46 172.67.75.39 TCP 54 51111 →
4 11:16:47.178243 192.168.0.46 172.67.75.39 TLSv1… 571 Client H
5 11:16:47.255959 172.67.75.39 192.168.0.46 TCP 60 443 → 51
6 11:16:47.267905 172.67.75.39 192.168.0.46 TLSv1… 1514 Server H
7 11:16:47.267905 172.67.75.39 192.168.0.46 TLSv1… 402 Applicat
8 11:16:47.267999 192.168.0.46 172.67.75.39 TCP 54 51111 →
9 11:16:49.048600 192.168.0.46 172.67.75.39 TLSv1… 118 Change C
10 11:16:49.048918 192.168.0.46 172.67.75.39 TLSv1… 146 Applicat
11 11:16:49.049309 192.168.0.46 172.67.75.39 TLSv1… 720 Applicat
12 11:16:49.073867 172.67.75.39 192.168.0.46 TCP 60 443 → 51
> Frame 1: 66 bytes on wire (528 bits), 66 byt
> Ethernet II, Src: Intel_a3:29:0b (34:f6:4b:a
> Internet Protocol Version 4, Src: 192.168.0.
> Transmission Control Protocol, Src Port: 511
0000 58 19 f8 d1 2e ae 34 f6 4b a3 29 0b 08
0010 00 34 6e a3 40 00 80 06 d3 df c0 a8 00
0020 4b 27 c7 a7 01 bb 0e 47 8a 2d 00 00 00
0030 fa f0 5a 07 00 00 02 04 05 b4 01 03 03
0040 04 02
Wireshark_Masterclass_Lesson1_Setup.pcapng
Packets: 89
Profile: Default


Wireshark_Masterclass_Lesson1_Setup.pcapng
Apply a display filter ... <⌘/>
Destination Protocol Lengt TCP Segment Len Info
6 172.67.75.39 TCP 66 0 51111 → 443 [SYN] Seq=0 W
9 192.168.0.46 TCP 66 0 443 → 51111 [SYN, ACK] Se
6 172.67.75.39 TCP 54 0 51111 → 443 [ACK] Seq=1 A
6 172.67.75.39 TLSv1… 571 517 Client Hello (SNI=www.wire
9 192.168.0.46 TCP 60 0 443 → 51111 [ACK] Seq=1 A
9 192.168.0.46 TLSv1… 1514 1460 Server Hello, Change Ciph
9 192.168.0.46 TLSv1… 402 348 Application Data
6 172.67.75.39 TCP 54 0 51111 → 443 [ACK] Seq=518
6 172.67.75.39 TLSv1… 118 64 Change Cipher Spec, Applic
6 172.67.75.39 TLSv1… 146 92 Application Data
6 172.67.75.39 TLSv1… 720 666 Application Data
9 192.168.0.46 TCP 60 0 443 → 51111 [ACK] Seq=180
> Frame 1: 66 bytes on wire (528 bits), 66 byt
> Ethernet II, Src: Intel_a3:29:0b (34:f6:4b:a
> Internet Protocol Version 4, Src: 192.168.0.
> Transmission Control Protocol, Src Port: 511
Source Port: 51111
Destination Port: 443
[Stream index: 0]
[Stream Packet Number: 1]
> [Conversation completeness: Incomplete, DA
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence
Sequence Number (raw): 239569453
[Next Sequence Number: 1 (relative seq
0000 58 19 f8 d1 2e ae 34 f6 4b a3 29 0b 08
0010 00 34 6e a3 40 00 80 06 d3 df c0 a8 00
0020 4b 27 c7 a7 01 bb 0e 47 8a 2d 00 00 00
0030 fa f0 5a 07 00 00 02 04 05 b4 01 03 03
0040 04 02
TCP Segment Len (tcp.len)
Packets: 89
Profile: Default

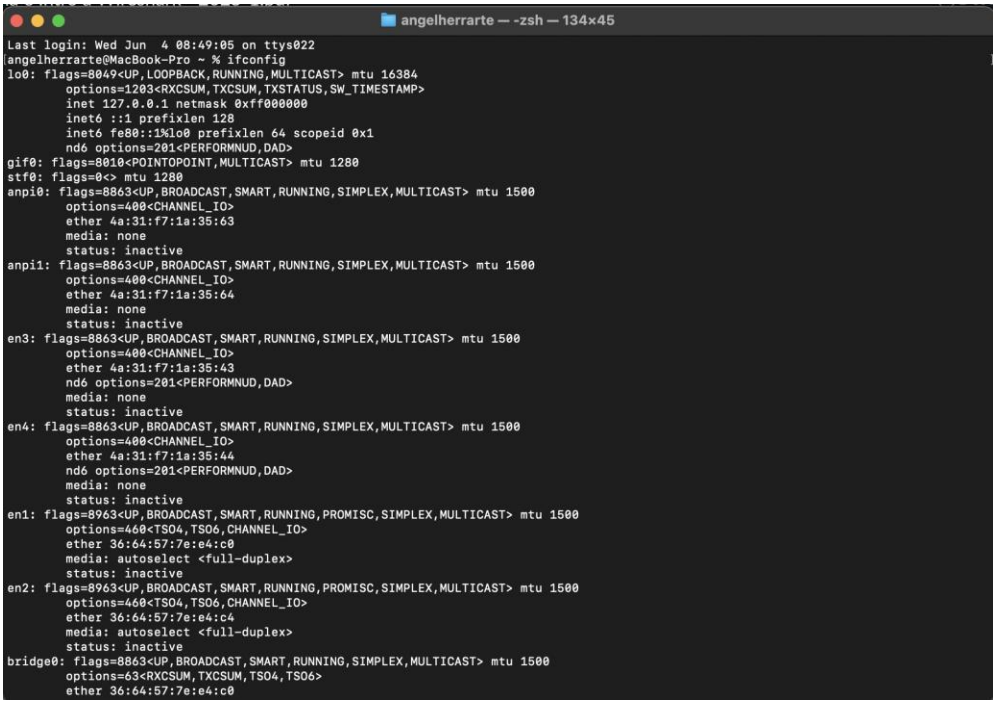

Wireshark_Masterclass_Lesson1_Setup.pcapng
Apply a display filter ... <⌘/>
Source | Destination | Protocol | TCP Segment Len | Info
Transmission Control Protocol
TCP Segment Len (tcp.len)
192.168.0.46 QUIC Protected Payload (KP0
172.67.75.39 QUIC Protected Payload (KP0
172.67.75.39 TCP 0 51111 → 443 [SYN] Seq=
192.168.0.46 TCP 0 443 → 51111 [SYN, ACK]
172.67.75.39 TCP 0 51111 → 443 [ACK] Seq=
192.168.0.46 TCP 0 443 → 51111 [ACK] Seq=
172.67.75.39 TCP 0 51111 → 443 [ACK] Seq=
192.168.0.46 TCP 0 443 → 51111 [ACK] Seq=
Frame 1: 66 bytes on wire (528 bits), 66 byt
Ethernet II, Src: Intel_a3:29:0b (34:f6:4b:a
Internet Protocol Version 4, Src: 192.168.0.
Transmission Control Protocol, Src Port: 511
Source Port: 51111
Destination Port: 443
[Stream index: 0]
[Stream Packet Number: 1]
[Conversation completeness: Incomplete, DA
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence
Sequence Number (raw): 239569453
0000 58 19 f8 d1 2e ae 34 f6 4b a3 29 0b 08
0010 00 34 6e a3 40 00 80 06 d3 df c0 a8 00
0020 4b 27 c7 a7 01 bb 0e 47 8a 2d 00 00 00
0030 fa f0 5a 07 00 00 02 04 05 b4 01 03 03
0040 04 02
TCP Segment Len (tcp.len)
Packets: 89
Profile: Default


Wireshark_Masterclass_Lesson1_Setup.pcapng
Apply a display filter ... <⌘/>
No. | Destination | Time | Source | Protocol | TCP Segment Len
85 192.168.0.46 11:16:49.645336 172.67.75.39 QUIC
86 172.67.75.39 11:16:49.645802 192.168.0.46 QUIC
87 172.67.75.39 11:16:49.646088 192.168.0.46 QUIC
88 192.168.0.46 11:16:49.651857 172.67.75.39 QUIC
89 192.168.0.46 11:16:49.651857 172.67.75.39 QUIC
1 172.67.75.39 11:16:47.126585 192.168.0.46 TCP
2 192.168.0.46 11:16:47.177831 172.67.75.39 TCP
3 172.67.75.39 11:16:47.177959 192.168.0.46 TCP
5 192.168.0.46 11:16:47.255959 172.67.75.39 TCP
8 172.67.75.39 11:16:47.267999 192.168.0.46 TCP
12 192.168.0.46 11:16:49.073867 172.67.75.39 TCP
Frame 1: 66 bytes on wire (528 bits), 66 byt
Ethernet II, Src: Intel_a3:29:0b (34:f6:4b:a
Internet Protocol Version 4, Src: 192.168.0.
Transmission Control Protocol, Src Port: 511
Source Port: 51111
Destination Port: 443
[Stream index: 0]
[Stream Packet Number: 1]
[Conversation completeness: Incomplete, DA
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence
Sequence Number (raw): 239569453
0000 58 19 f8 d1 2e ae 34 f6 4b a3 29 0b 08
0010 00 34 6e a3 40 00 80 06 d3 df c0 a8 00
0020 4b 27 c7 a7 01 bb 0e 47 8a 2d 00 00 00
0030 fa f0 5a 07 00 00 02 04 05 b4 01 03 03
0040 04 02


Wireshark_Masterclass_Lesson1_Setup.pcapng
tcp.flags & 0x02 || tcp.flags.syn == 1
No. Destination Time Source Protocol TCP Segment Len
1 172.67.75.39 11:16:47.126585 192.168.0.46 TCP
2 192.168.0.46 11:16:47.177831 172.67.75.39 TCP


Wireshark_Masterclass_Lesson1_Setup.pcapng
tcp.flags & 0x02 || tcp.flags.syn == 1
No. Time Source Destination Protocol Length Info
2 0.051246 172.67.75.39 192.168.0.46 TCP 66 443 → 51111…
1 0.000000 192.168.0.46 172.67.75.39 TCP 66 51111 → 443…
Frame 2: 66 bytes on wire (528 bits), 66 byt
Ethernet II, Src: Commscope_d1:2e:ae (58:19:
Internet Protocol Version 4, Src: 172.67.75.
Transmission Control Protocol, Src Port: 443
Source Port: 443
Destination Port: 51111
[Stream index: 0]
[Stream Packet Number: 2]
[Conversation completeness: Incomplete, DA
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence
Sequence Number (raw): 3298756742
0000 34 f6 4b a3 29 0b 58 19 f8 d1 2e ae 08
0010 00 34 00 00 40 00 3b 06 87 83 ac 43 4b
0020 00 2e 01 bb c7 a7 c4 9f 08 86 0e 47 8a
0030 ff ff 87 fb 00 00 02 04 05 78 01 01 04
0040 03 0a

Parte 2



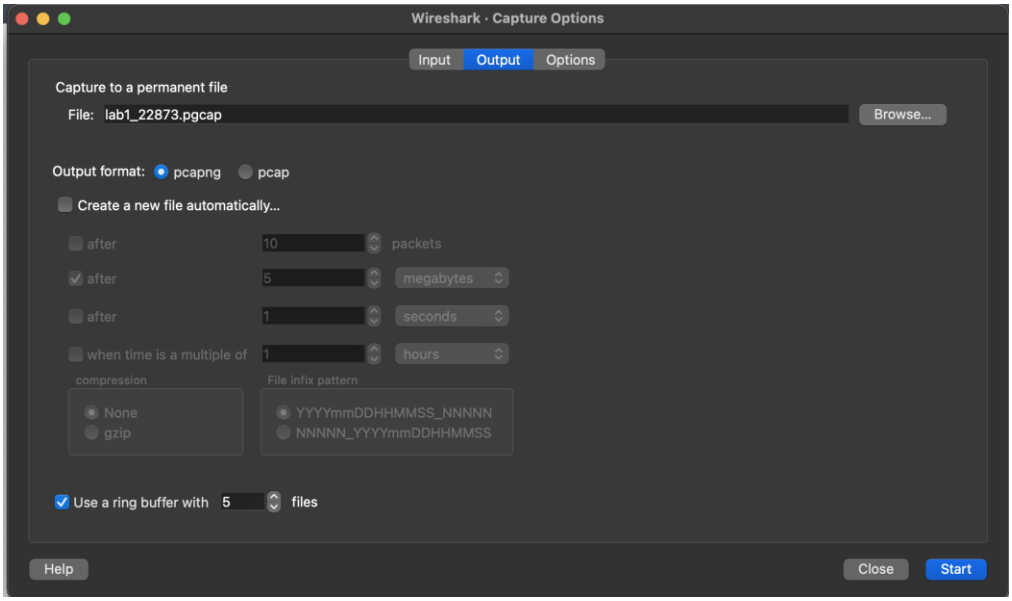
Interfaces principales identificadas:

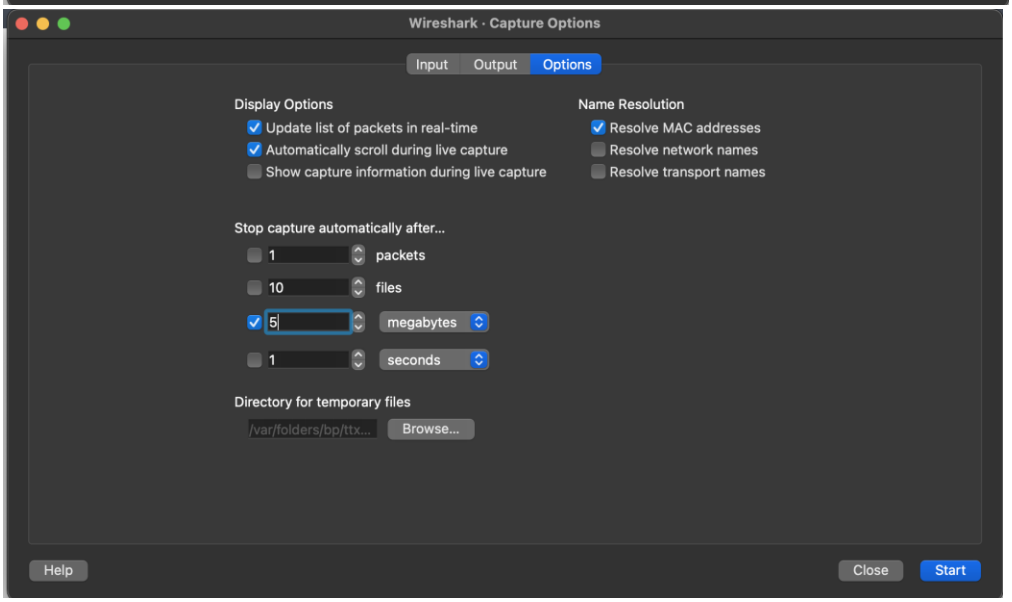
- **lo0**: Interfaz de loopback (127.0.0.1) - comunicación interna del sistema
- **en0**: Interfaz WiFi principal - **ACTIVA** con IP 192.168.0.16
- **awdl0**: Apple Wireless Direct Link (AirDrop, Handoff)
- **utun0-utun3**: Interfaces de túnel VPN
- **bridge0, en1, en2**: Interfaces bridge y Ethernet virtuales (inactivas)

¿Cuál es su interfaz de red?

La interfaz de red principal es en0 porque:

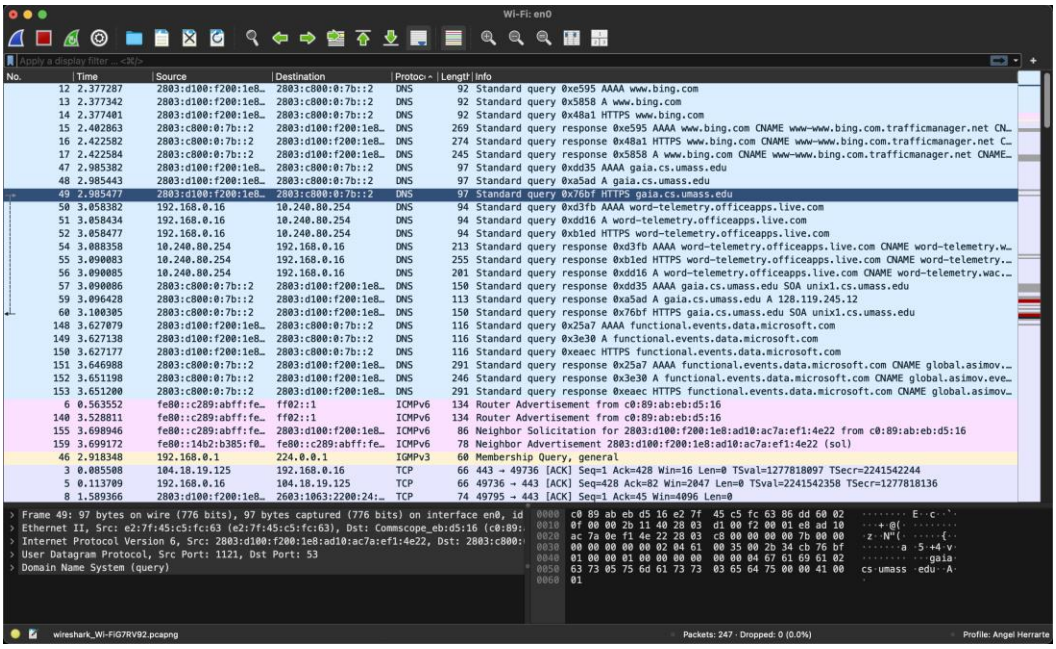
- Status: **active** (las demás están inactive)
- Tiene IP asignada: **192.168.0.16**
- Es la interfaz WiFi que está conectada a la red

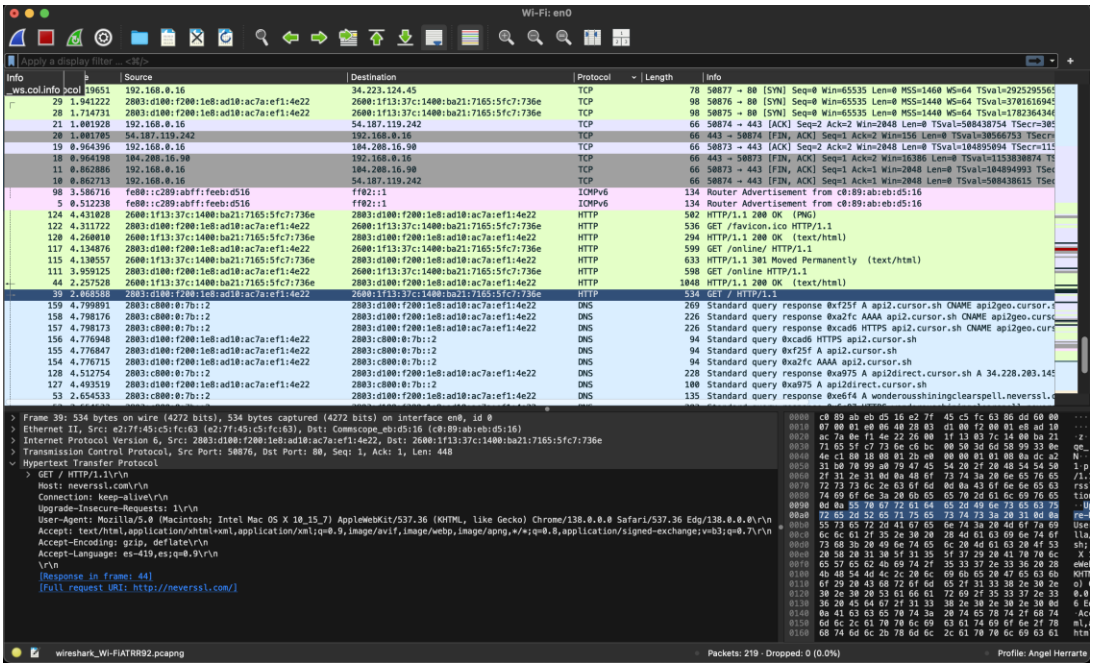

Wireshark · Capture Options
Input
Output
Options
Capture to a permanent file
File: lab1_22873.pgcap
Browse...
Output format: pcapng pcap
Create a new file automatically...
after 10 packets
after 5 megabytes
after 1 seconds
when time is a multiple of 1 hours
compression
None
gzip
File infix pattern
YYYYmmDDHHMMSS_NNNNN
NNNNN_YYYYmmDDHHMMSS
Use a ring buffer with 5 files
Help
Close
Start

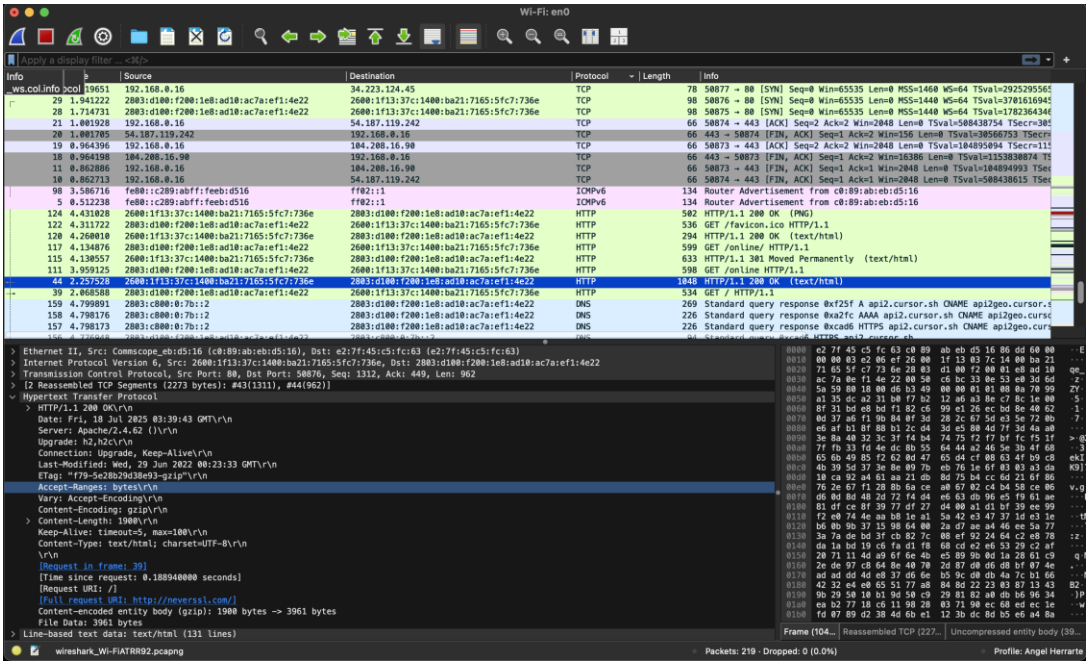

Wireshark · Capture Options
Input
Output
Options
Display Options
Update list of packets in real-time
Automatically scroll during live capture
Show capture information during live capture
Name Resolution
Resolve MAC addresses
Resolve network names
Resolve transport names
Stop capture automatically after...
1 packets
10 files
5 megabytes
1 seconds
Directory for temporary files
/var/folders/bp/ttx...
Browse...
Help
Close
Start

https://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html

Congratulations! You've downloaded the first Wireshark lab file!







Para esta sección utilicé la página http://neverssl.com ya que la anterior no soportaba http

¿Qué versión de HTTP está ejecutando su navegador?

HTTP/1.1 (visible en el paquete 39: "GET / HTTP/1.1")

¿Qué versión de HTTP está ejecutando el servidor?

- **HTTP/1.1** (visible en "HTTP/1.1 200 OK" en la respuesta)
- **Servidor:** Apache/2.4.62 (visible en el header "Server:")

¿Qué lenguajes (si aplica) indica el navegador que acepta a el servidor?

es-419,es;q=0.9 (español de Latinoamérica y español general)

¿Cuántos bytes de contenido fueron devueltos por el servidor?

- **1900 bytes** (visible en "Content-Length: 1900")
- **Nota:** El contenido está comprimido con gzip (1900 bytes comprimidos → 3961 bytes descomprimidos)

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

- **Cliente (computadora):** Para problemas de conectividad local, DNS, o procesamiento del navegador
- **Router/Switch intermedio:** Para problemas de latencia de red, pérdida de paquetes, o congestión
- **Servidor: SÍ es conveniente** instalar Wireshark porque permite:
  Distinguir si el problema es procesamiento del servidor vs. problemas de red
  Analizar si el servidor está recibiendo las peticiones correctamente
  Identificar problemas de configuración del servidor

Discusión sobre la actividad, su experiencia y hallazgos

Parte 1 - Red Humana

El código Morse resultó más intuitivo pero propenso a errores por la variabilidad de duración. El código de Baudot fue más preciso una vez memorizado. La transmisión empaquetada introdujo latencia y necesidad de protocolos de control. El sistema de conmutación demostró ventajas de centralización, pero creó puntos de falla.

Parte 2 - Wireshark

La personalización de Wireshark y configuración del ring buffer proporcionó comprensión práctica del monitoreo de redes. La migración de HTTP a HTTPS (observada en gaia.cs.umass.edu a neverssl) evidencia la evolución hacia mayor seguridad. El análisis reveló estructuras detalladas de comunicación web, incluyendo compresión gzip y keep-alive.

Comentarios

- La transición HTTP→HTTPS refleja la priorización de seguridad en redes modernas
- Los códigos históricos proporcionan perspectiva importante sobre fundamentos de comunicación
- Wireshark es una herramienta esencial para diagnóstico profesional de redes

Conclusiones

El laboratorio cumplió perfectamente con sus objetivos al combinar fundamentos históricos con herramientas modernas. La experiencia práctica con protocolos de transmisión y análisis de paquetes fortaleció la comprensión teórica de redes de computadoras y su evolución hacia sistemas más seguros y eficientes.

Referencias Utilizadas

Kurose, J. F., & Ross, K. W. (2021). *Computer networking: A top-down approach* (8th ed.). Pearson Education.

Wireshark Foundation. (2023). *Wireshark user's guide: Network protocol analyzer documentation* (Version 4.0). https://www.wireshark.org/docs/wsug_html_chunked/