Universidad Diego Portales
Facultad de Ingeniería y Ciencias

Informe de Tarea 1

Taller de Redes y Servicios

Autores:
Estudiantes: Ezequiel Morales ; Francisco Piñera
RUT: 22602790-4 ; 21667648-3

Carrera: Ingeniería Civil en Informatica y Telecomunicaciones
Email 1: ezequiel.morales@mail.udp.cl
Email 2: francisco.pinera@mail.udp.cl

Segundo Semestre - 29 septiembre de 2025

Índice

1. Introducción

El objetivo principal de este trabajo será obtener una captura de tráfico extensa utilizando el programa en Wireshark y guardarla para su posterior análisis, donde se buscara encontrar patrones de tráfico específicos utilizando los filtros de captura propios de la aplicación y en base a protocolos de red más específicos. Se espera que lo mínimo obtenible sea una captura de 8 horas con una cantidad de paquetes acorde a la duración de la captura. Junto con ello, se espera que al final del análisis se hayan podido encontrar 10 patrones de trafico, que sean fácilmente reconocibles. Y aparte de eso, analizar cada patrón posterior para así poder detectar anomalías y de paso explicar causas de porque el patrón detectado se ve de esa forma.

El destino de analizar patrones de trafico radica en como el administrador utiliza su capacidad de análisis para encontrar mínimos comunes entre cientos de miles de consultas que se pueden llegar a encontrar en una sola captura de trafico. Es por ello que analizar críticamente el comportamiento de ciertos patrones de trafico sera fundamental para comparar con otras personas y así encontrar explicaciones coherentes que justifiquen la causa real de los comportamientos vistos en cada patrón.

2. Metodología

2.1. Resumen de las herramientas

Las herramientas utilizadas para realizar la captura extensa son:

- VirtualBox
- Linux Mint Cinnamon 22.1 64-Bit
- Wireshark para Linux de 64-Bit

2.2. Levantamiento de la red LAN del hogar

A continuación se presenta la tabla con los datos de todos los dispositivos que participaron en la captura de trafico

	PC sniffer	Televisor	telefono 1
Marca	lenovo	samsung	samsung
Modelo	15IAH7	Q60AA	SM-G970F
Sistema operativo	windows 11	Tizen OS	android 12
Kernel/firmware	10.0.26100	2220.9.0	4.14.113
IP	192.168.1.114	192.168.1.106	192.168.1.112 y 154
MAC	FA:6A:DD:8E:39:F9	7C:0A:3F:71:CA:3F	A8:DB:03:13:3A:79

telefono 2	telefono 3	telefono 4	telefono 5
samsung	samsung	xiaomi	motorola
SM-A556E	SM-G998B	2201117SL	moto g13
android 15	andorid 15	hyper OS	android 14
6.1.93	5.4.242	4.14.186	UHAS34.29-20
192.168.1.133	192.168.1.147	192.168.1.161	192.168.1.189
00:2B:70:F8:C7:5D	F4:02:28:D0:01:7C	E4:84:D3:8F:73:64	40:FA:FE:35:6A:8B

switch 2
Nintendo
switch 2
Horizon OS
20.4.0
192.168.1.143
3C:A9:AB:0B:C1:92

Consideraciones importantes:

- Dado que la maquina virtual debe estar configurada para utilizar el internet de la red LAN, entonces eso implica que la maquina virtual tenga su propia dirección IP aparte de la dirección IP propia del pc. Es por ello que la dirección IP utilizada durante la captura fue la 192.168.1.152

2.3. Topología de red del hogar

A continuación se presenta gráficamente la topología de red del hogar. Esta topología fue construida en draw.io

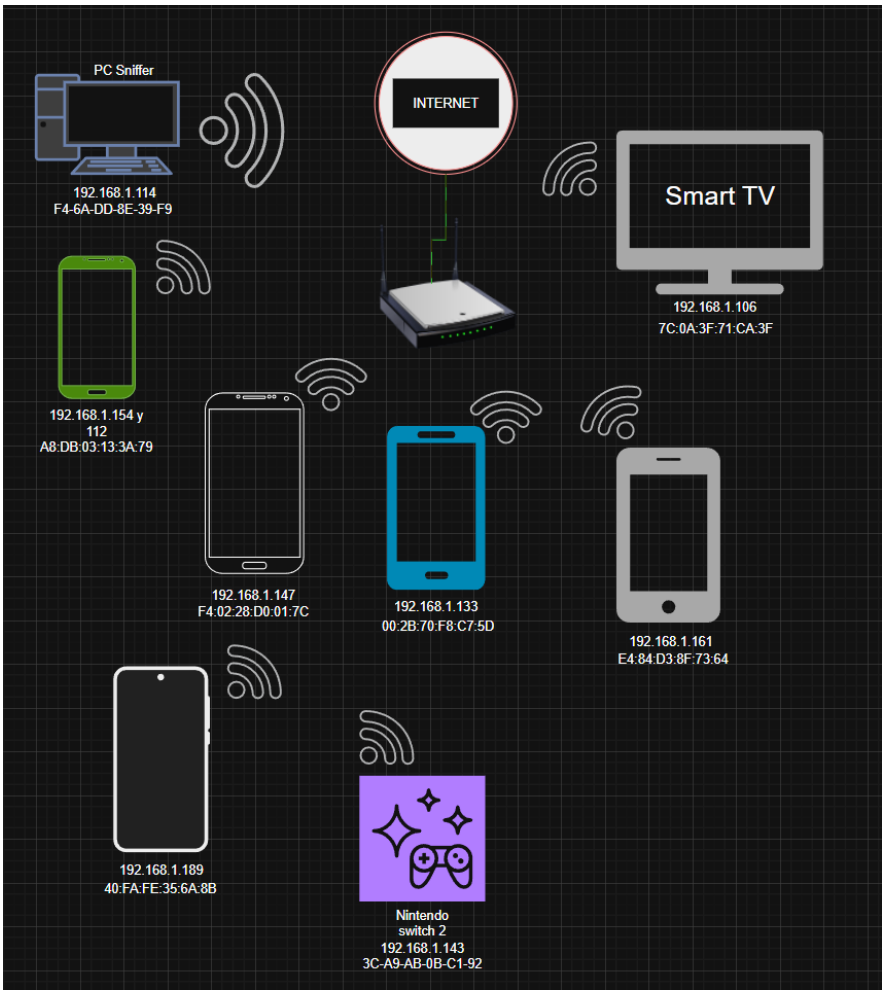

Figura 1: Topología del hogar

3. Desarrollo

La captura tuvo una duración exacta de 9 horas, por lo que una vez finalizada, se obtuvieron un total exacto de 200917 paquetes.

Inmediatamente después de finalizar la captura se comenzó a realizar la obtención de patrones mediante los filtros de captura de wireshark.

Como configuración preliminar para los filtros, se ordeno la lista de paquetes desde el numero 1 hasta el ultimo de ellos, y también se ordenaron los tiempos de cada paquete en relación al paquete anterior, dejando así un tiempo que dicta cuantos segundos pasaron desde el ultimo paquete mostrado en la lista.

Como descripción general de la metodología para encontrar patrones, primero se revisa

la jerarquía de protocolos de red, para así encontrar un protocolo que cumpla con tener una cantidad de paquetes aceptables y que no sea uno de la lista prohibida señalada en el enunciado. Después se agrega ese protocolo al filtro, paso seguido se filtra por alguna dirección ip fuente y después por una ip destino si es necesario. Después se observa si la dirección adonde se envía el paquete es destacable y ya a partir de ahí los filtros serán mas específicos dependiendo de que características especiales tenga cada paquete.

3.1. Protocolo DNS

- **DNS con IP origen finalizada en 152** Primero se hizo un filtro con la dirección ip de origen 192.168.1.152. Dado que solo había una ip de destino asociada a esa ip no fue necesario aplicar tal filtro. Después se observo que un tipo de query asociada se repetía constantemente, la cual es çonnectivity-check-ubuntu type A", por lo que se agrego al filtro con el operador 'y'. Como resultado final se obtuvieron 94 paquetes.

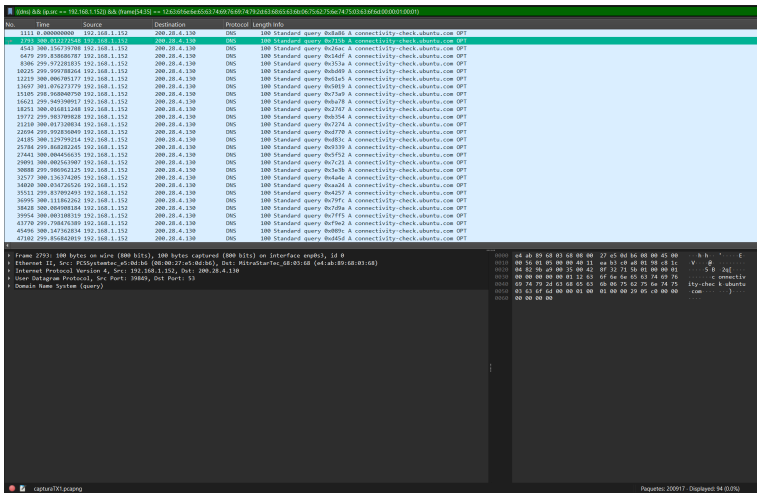

Figura 2: DNS

3.2. Protocolo DHCP

- **DHCP con IP origen 0000** Primero se utilizo como filtro la ip de origen 0.0.0.0, la cual corresponde a un equipo que no cuenta con dirección ip en ese momento. Luego es notable que no hay mucho mas de donde sostenerse pero fijándonos bien, vemos que una característica reconocible del protocolo es el tipo de mensaje DHCP, por lo que se elige filtrar por el tipo de mensaje DISCOVER el cual esta marcado en el id opción 1. Una vez hecho esto, se obtiene un total de 83 paquetes.

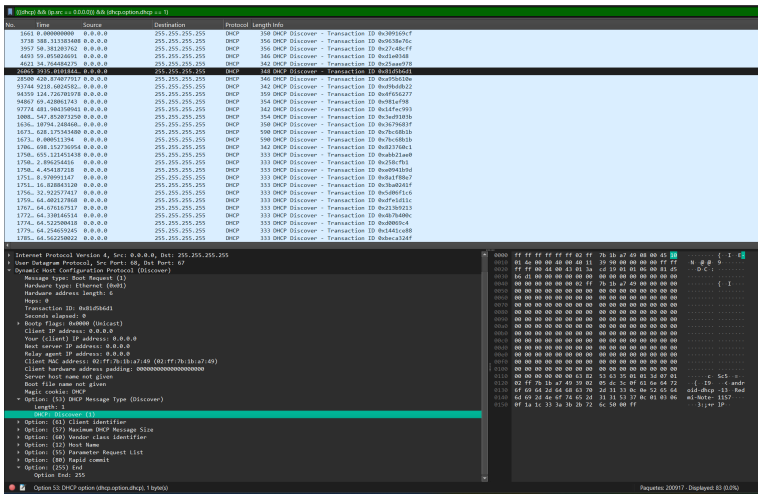

Figura 3: DHCP 0000

- **DHCP con IP origen terminada en 152** Nuevamente se escoge como filtro la ip de origen terminada en 152. Luego, para esta ocasión no se filtro directamente por un mensaje DHCP especifico porque por descarte ya se sabe que esta dirección ip esta intentando renovar su concesión con el servidor DHCP por lo que el tipo de mensaje esperable es REQUEST. Se obtuvieron un total de 179 paquetes.

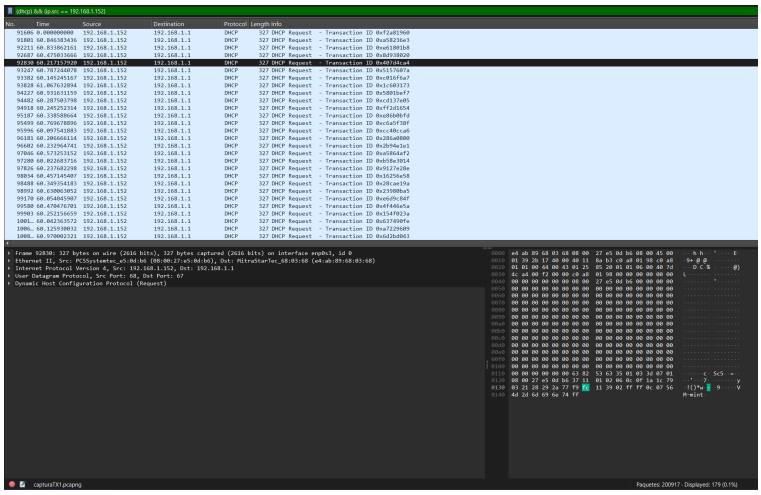

Figura 4: DHCP 152

3.3. Protocolo HTTP

- **HTTP con ip origen terminada en 152** Volvemos a aplicar la ip origen como filtro y se aprieta ENTER. Dado que el mayor distintivo esta en la informacion de destino, observamos que hay un tipo de mensaje HTTP el cual es de tipo GET. Por lo que

nos dirigimos al espacio de HTP y aplicamos como filtro la informacion marcada como 'GET/HTTP/1.1'. Como resultado se obtuvieron 94 paquetes.

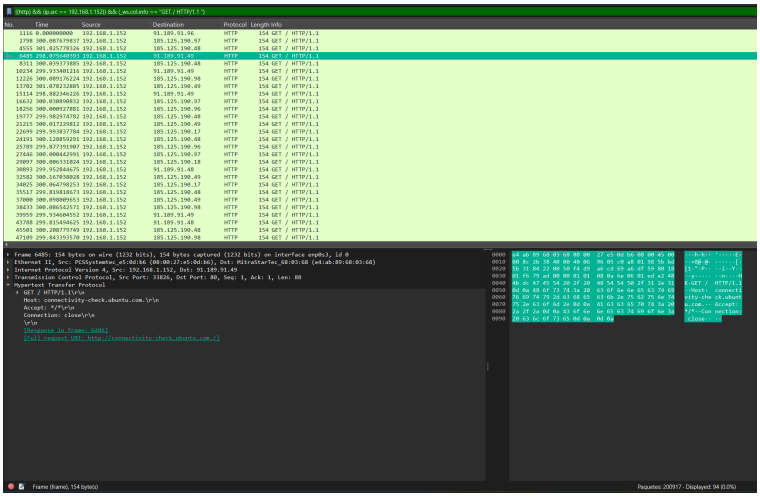

Figura 5: HTTP

3.4. Protocolo MDNS

- **MDNS con ip origen terminada en 106** Ahora se decide aplicar la dirección ip origen 192.168.1.106, la cual corresponde a la tele. Dado que todos los destinos son la ip 224.0.0.251, no es necesario aplicarlo como filtro. Como paso seguido se observa que muchas consultas son de tipo response las cuales van hacia spotify, y dado que la televisión estuvo mucho tiempo reproduciendo spotify, entonces se escoge esa consulta como filtro de captura. Se obtuvieron un total de 1213 paquetes.

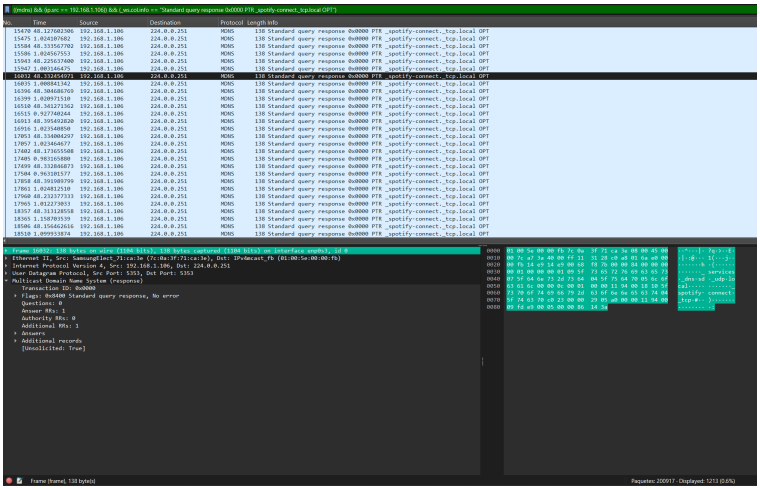

Figura 6: MDNS

3.5. Protocolo SSDP

- **SSDP con ip origen 1.1.1.2** Se decidio aplicar como filtro la dirección ip 1.1.1.2, y dado que las ip de destino son solo 239.255.255.250, no es necesario aplicarlo al filtro. Del mismo modo, todas las consultas poseen exactamente el mismo destino e información por lo que el filtro se mantiene tal cual como esta. Se obtuvieron un total de 304 paquetes

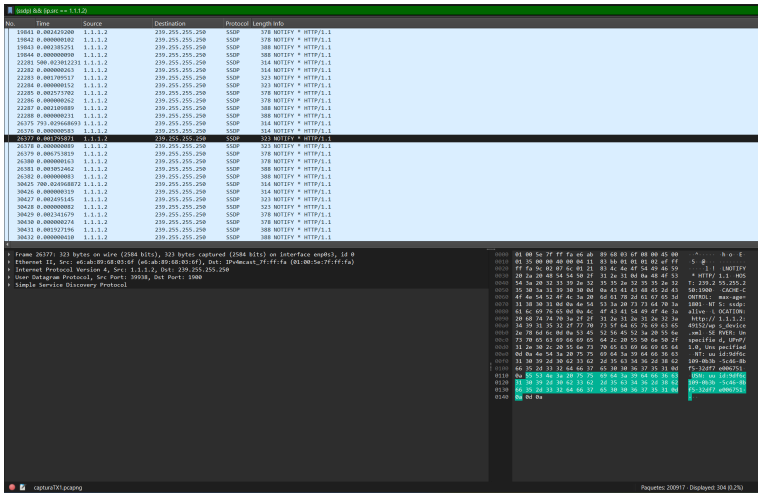

Figura 7: SSDP 1112

- **SSDP con ip origen terminada en 114** Para la segunda captura ssdp se decidió utilizar como filtro la ip de origen 192.168.1.114, la cual corresponde a uno de los escasos paquetes capturados del PC utilizado como sniffer. Dado que la ip de destino y la info de destino es la siempre la misma, no se aplican al filtro de captura. Por ende, siendo que no había información relevante para filtrar mas paquetes y reducir la cantidad, se decidió aplicar como filtro que solo estuvieran aquellos paquetes con un tamaño de frame de 212 bytes. El resultado final fueron 1092 paquetes.

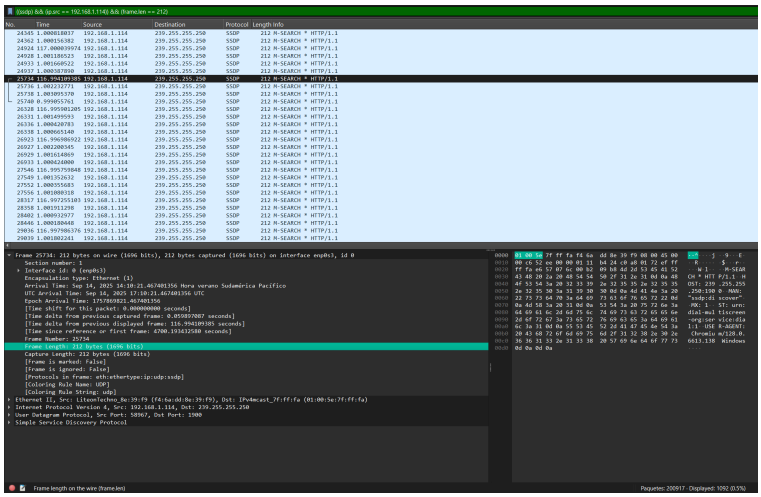

Figura 8: SSDP 114

3.6. Protocolo TLS

- **TLS con ip de origen terminada en 152** Para este patron nuevamente se opto por utilizar la ip terminada en 152. Dado que las direcciones IP de destino variaban mucho esta vez, se decidio utilizar como filtro la ip 35.186.224.46. Aun estaba la opcion de aplicar el filtro de la informacion de destino pero como ya quedaban solo 28 paquetes se opto por dejarlo asi nada mas para evitar quedar con muy pocos paquetes. Es por ello que el resultado final fueron 28 paquetes.

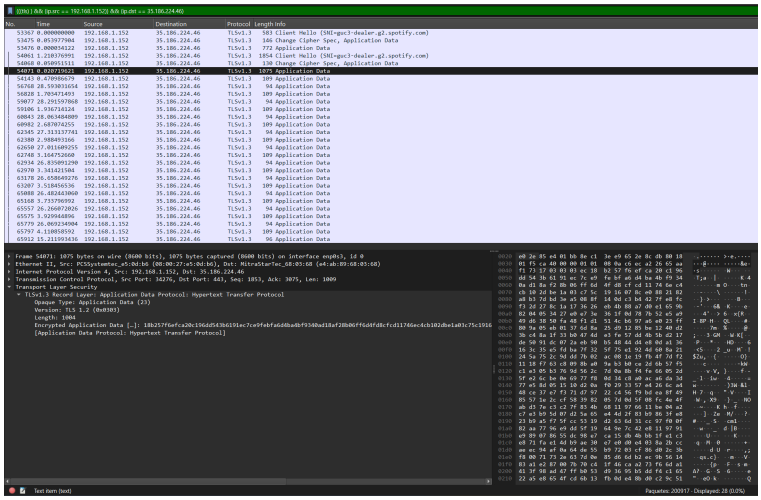

Figura 9: TLS

3.7. Protocolo IGMP

- **IGMP con ip de origen terminada en 1** Para el protocolo IGMP se eligió utilizar como origen la dirección ip 192.168.1.1. Al inicio fue complicado encontrar un patrón con este protocolo a pesar de que se capturaron varios paquetes, pero al decidir seleccionar como filtro la opción de que solo se muestren aquello paquetes que tienen la información de destino 'Membership query, general', se logro encontrar un patrón muy bien definido. Como resultado se obtuvieron 241 paquetes.

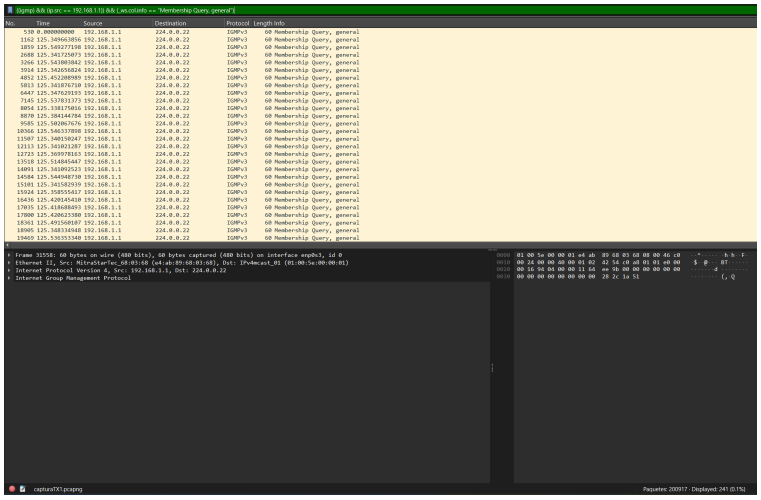

Figura 10: IGMP

3.8. Protocolo NTP

- **NTP con la ip de origen 185.125.190.57** Para este protocolo en particular, con solo 32 paquetes en toda la captura, se logro encontrar un patrón definido solo filtrando con el protocolo. No obstante, así tal cual el filtro es muy poco pulido por lo que se decidio utilizar como filtro mostrar aquellos paquetes que partieran desde el numero 10009. Se pudo haber dejado así tal cual pero dado que el filtro mostraba los paquetes de ida y vuelta, eso implicaba tener un patrón poco pulido, por lo que se opto por aplicar como filtro utilizar la ip de origen señalada arriba, la cual tiene como destino solo la ip 192.168.1.152. Ya con esto el patrón quedo mucho mejor definido y mas preciso. El resultado final fueron 14 paquetes.

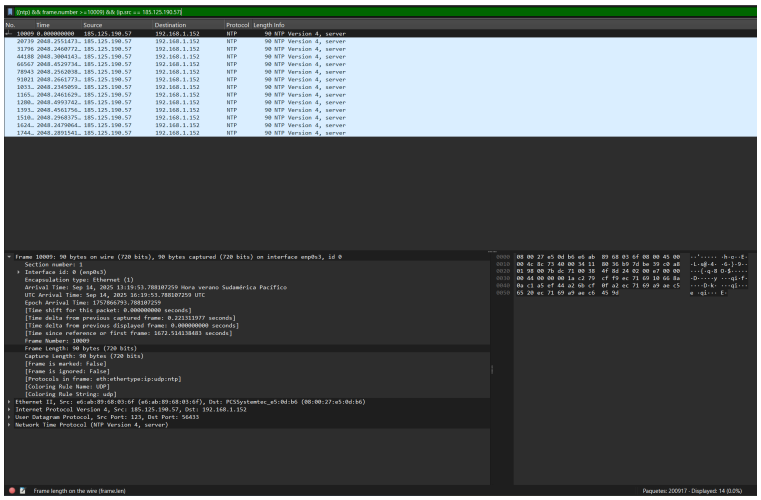

Figura 11: NTP

Ya como resumen de lo mostrado aquí, a continuación se muestra una tabla con todos los filtros aplicados

N Patrón	Filtro de captura
1	((dns) && (ip.src == 192.168.1.152)) && (frame[54:35] == XXXXX)
2	(dhcp) && (ip.src == 192.168.1.152)
3	(((dhcp) && (ip.src == 0.0.0.0))) && (dhcp.option.dhcp == 1)
4	(ssdp) && (ip.src == 1.1.1.2)
5	((ssdp) && (ip.src == 192.168.1.114)) && (frame.len == 212)
6	((ntp) && frame.number ¿=10009) && (ip.src == 185.125.190.57)
7	((igmp) && (ip.src == 192.168.1.1)) && (_ws.col.info == "Membership Query, general")
8	((tls) && (ip.src == 192.168.1.152)) && (ip.dst == 35.186.224.46)
9	((mdns) (ip.src == 192.168.1.106)) ($_{w}s.col.info$ == "$Standardqueryresponse0x0000PTR_spotify - connect._tcp.localOPT$")
10	((http) (ip.src == 192.168.1.152)) ($_{w}s.col.info$ == "$GET/HTTP/1,1$")

Tabla 1: Filtros de captura para los 10 patrones

3.9. Captura aplicación: Youtube

Tras haber reconocido 10 patrones en la captura de aproximademente 9 horas, se procede a realizar una captura de paquetes con `Wireshark` a `Youtube` de aproximadmente 1 hora. En esta captura se obtuvieron 162.919 paquetes de los cuales los principales protocolos encontrados y utilizados por esta aplicación son **DNS, TLSv1.3, TCP y QUIC** además de encontrar un patrón en uno de estos, especificamente **DNS**.

Primeramente para encontrar este patrón se aplica el filtro de `DNS` con la ip de destino `192.168.136.129` de los cuales se encontró un patrón en la query especifica a **Standard query 0x**** A www.youtube.com OPT**, como resultado se obtienen 56 paquetes.

Filtro: `(dns) && (ip.dst == 192.168.136.2) && (frame[56:21] == 03:77:77:77:07:79:6f:75:74:75:62:65:03:63:6f:6d:00:00:01:00:01)`

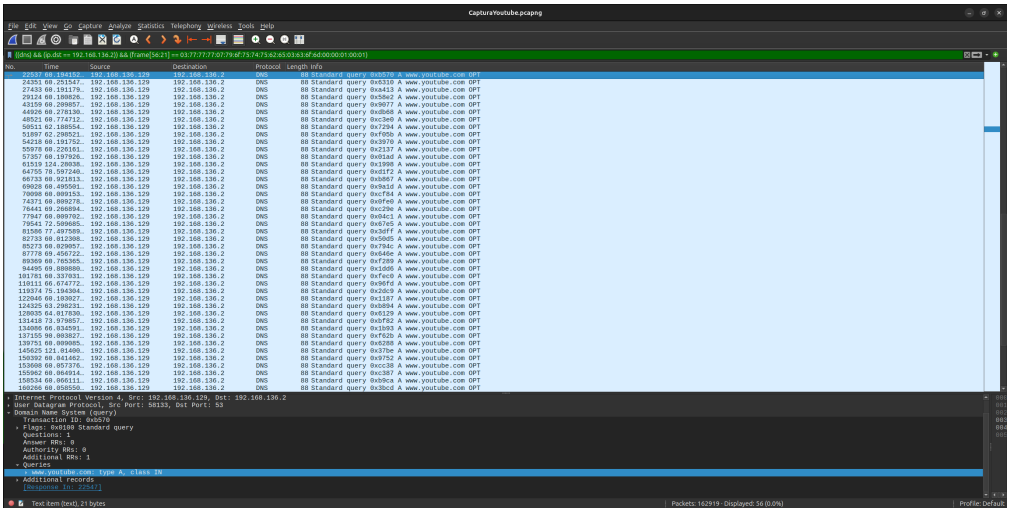

Figura 12: Filtro a querie `www.youtube.com`

Luego se procede a crear el gráfico representativo de este patrón encontrado.

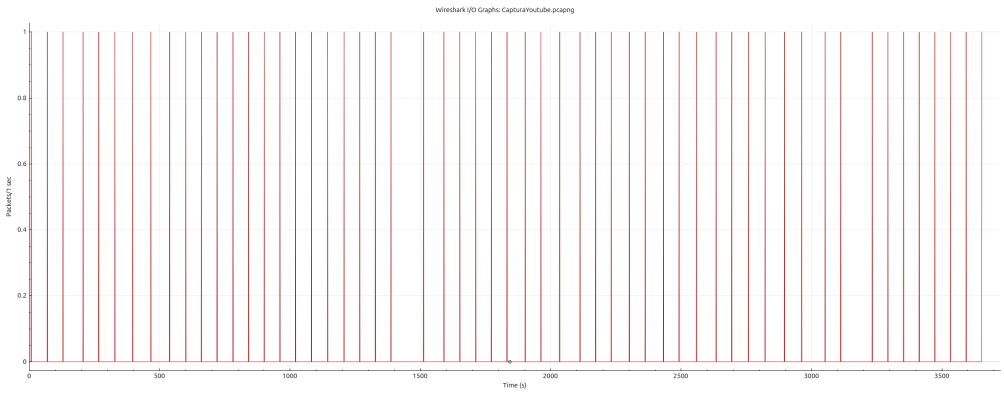

Figura 13: Grafico representativo patrón **DNS** a `www.youtube.com`

4. Análisis de resultados

4.1. Protocolo DNS

Observando los datos visibles de la lista de paquetes y la captura de la gráfica, se llega a la conclusión de que el patrón capturado cumple con las especificaciones porque todos los paquetes mostrados en la gráfica son uniformes con 1 solo paquete por consulta y tienen una separación temporal casi exacta de 300 segundos.

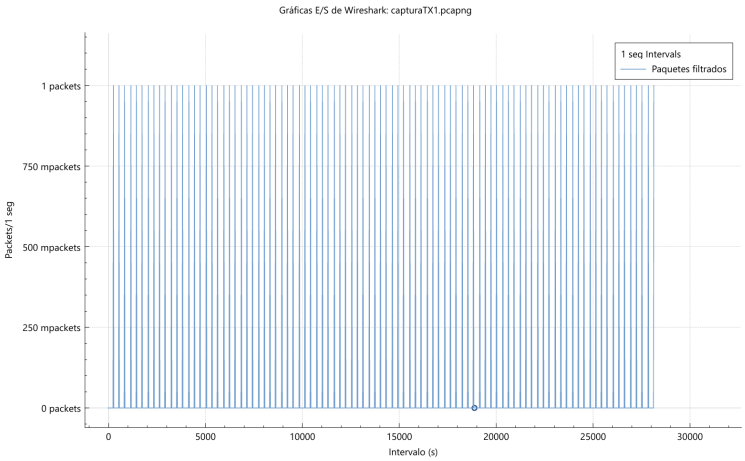

Figura 14: Grafica DNS

4.2. Protocolo DHCP

- **DHCP 0000** A pesar de que la gráfica muestra un desorden durante todo el espacio temporal, este patrón es valido porque al final de la gráfica se concentran mas del 60 por ciento de los paquetes que cumplen con el patron, los cuales son de 1 unidad y tienen una separación exacta de 64 segundos entre si.

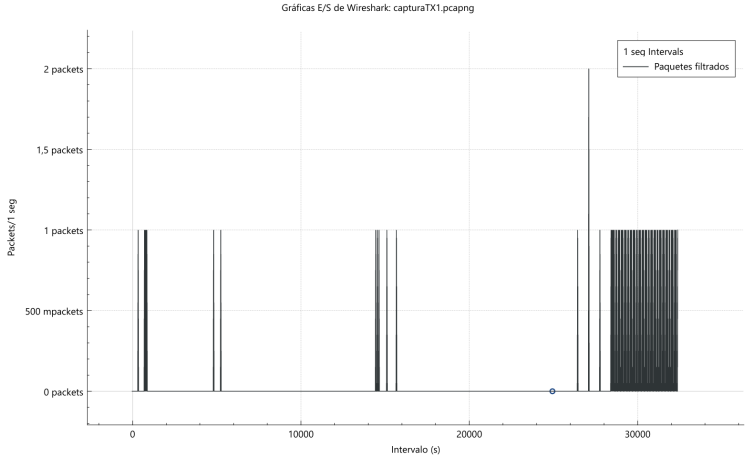

Figura 15: Grafica DHCP 0000

- **DHCP 152** Este patrón es valido porque es completamente uniforme, cada consulta contiene 1 paquete y la separación temporal es de exactamente 60 segundos.

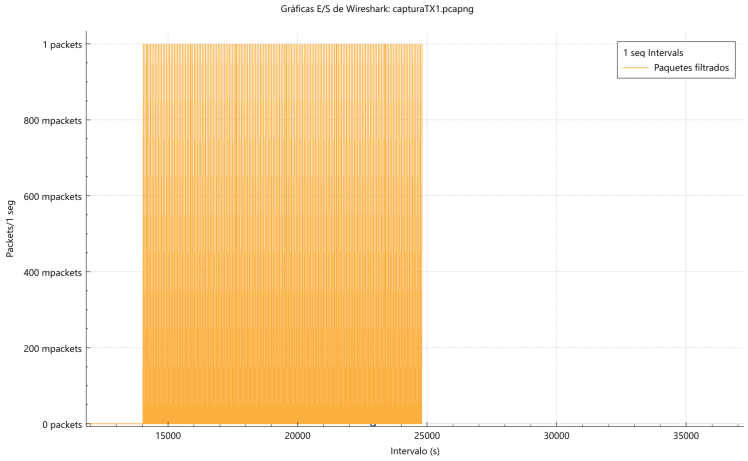

Figura 16: Grafica DHCP 152

4.3. Protocolo HTTP

Este patrón es valido porque es completamente uniforme, cada consulta contiene 1 paquetes y la separación temporal entre cada paquete es de exactamente 300 segundos.

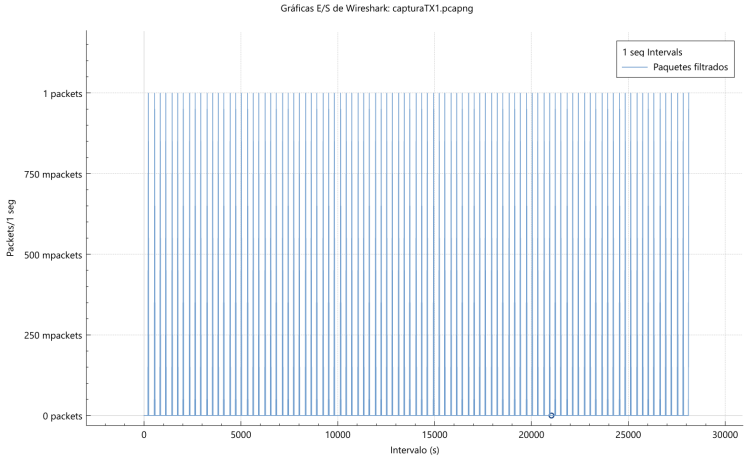

Figura 17: Gráfica HTTP

4.4. Protocolo IGMP

Este patrón a pesar de los saltos visibles que hay en la gráfica, es valido porque la mayoría del tiempo es completamente uniforme, con 1 paquete por consulta, y con una separación temporal exacta de 125 segundos.

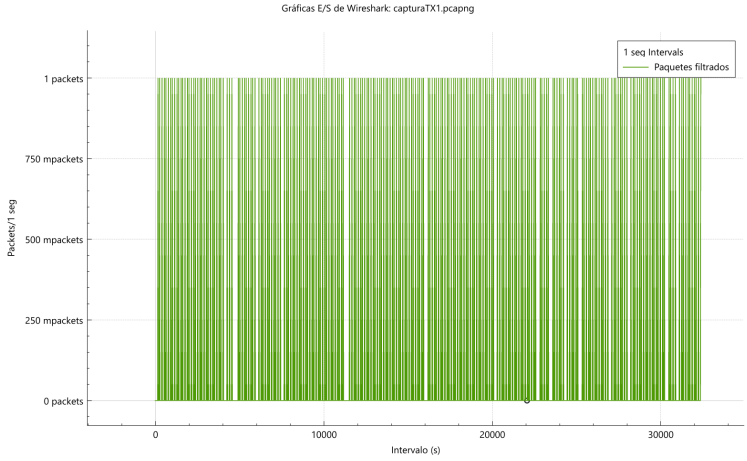

Figura 18: Gráfica IGMP

4.5. Protocolo MDNS

Los rasgos a destacar en este gráfico son los momentos en los cuales se envían 2 paquetes, y el momento justo en el cual hay un vació enorme en la gráfica. Exceptuando esos dos detalles, el patrón es completamente valido ya que en mas del 90 por ciento del tiempo el patrón es completamente uniforme, con solo 1 paquete por consulta compuesto por 2 envíos

debido a un retardo de 1 segundo en la secuencia de respuesta y con una separación temporal uniforme aproximada de 48 segundos + 1 extra por el retardo.

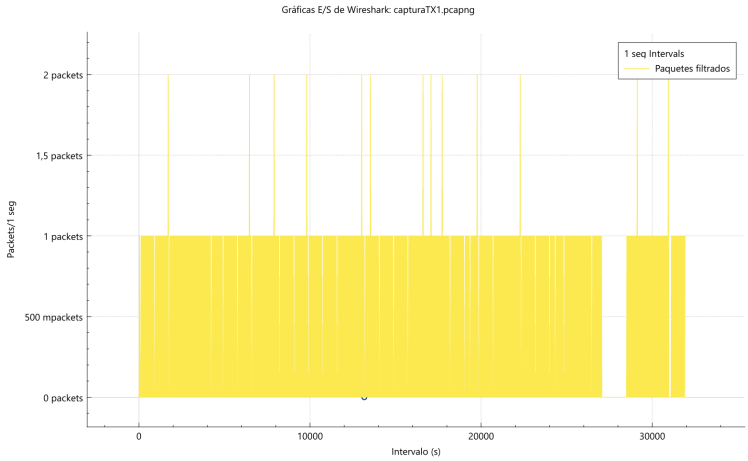

Figura 19: Gráfica MDNS

4.6. Protocolo NTP

A pesar de que este patrón solo tiene 14 paquetes, es valido porque es completamente uniforme, con 1 paquete enviado por consulta y con una separación temporal exacta de 2048 segundos.

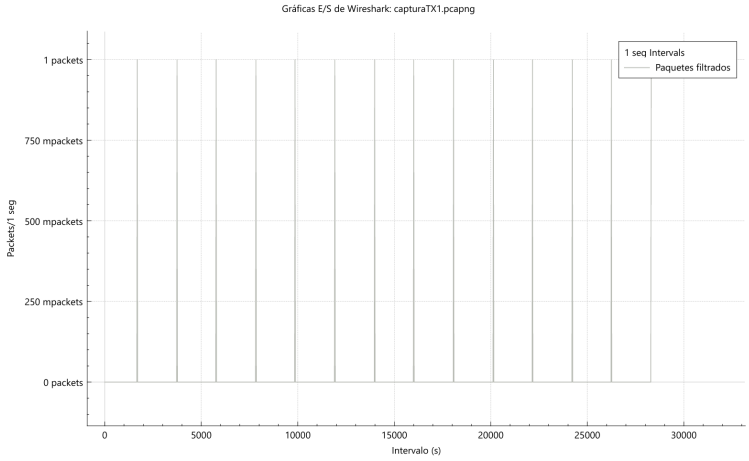

Figura 20: Grafica NTP

4.7. Protocolo SSDP

- **SSDP 1.1.1.2** Este patrón es valido porque en los segmentos donde la gráfica es uniforme y no hay grandes vacíos en la linea temporal, se envían un total de 8 paquetes

con una separación temporal variable pero estable que varia entre los 500 y los 800 segundos. Cabe mencionar que mas del 60 por ciento de los paquetes se encuentran en estos 2 segmentos reconocibles.

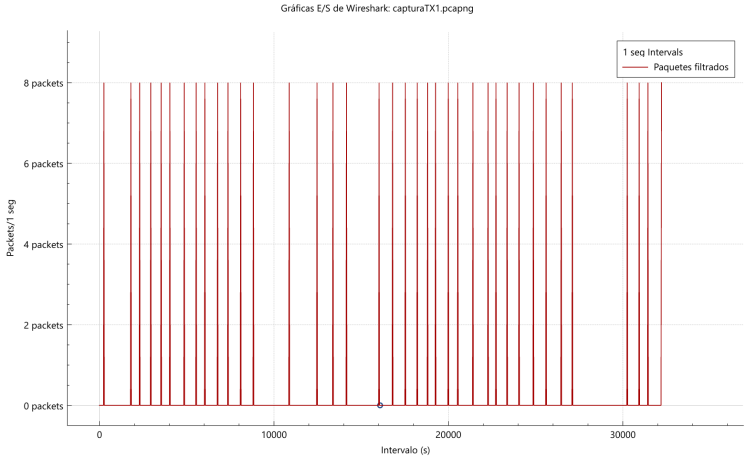

Figura 21: Grafica SSDP 1112

- **SSDP 114** Este patrón es valido porque es completamente uniforme, con 1 paquete por consulta + 3 paquetes extras de 1 segundo cada una debido al retardo general en el envió esperado del paquete, y con una separación temporal entre paquetes de exactamente 116 segundos.

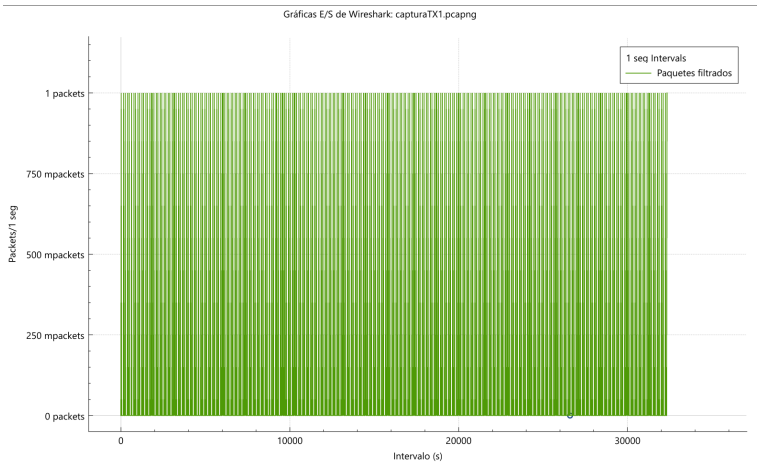

Figura 22: Grafica SSDP 114

4.8. Protocolo TLS

Este patrón es valido porque ignorando los primeros 8 paquetes que no representan a la gráfica, se obtiene un patrón uniforme de 1 paquete por consulta + 1 paquete extra por

retardo de 1 segundo, y con una separación temporal entre paquetes de aproximadamente 26 segundos. Cabe mencionar que los paquetes que forman parte del patrón son 20 de 28 totales.

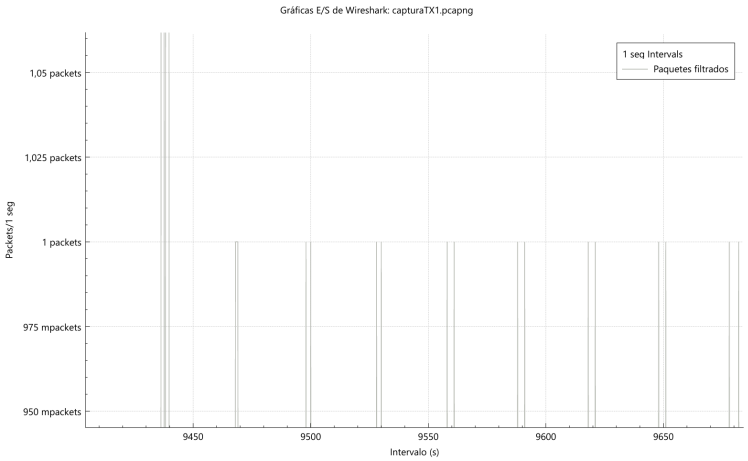

Figura 23: Grafica TLS

Ya para finalizar, a continuación se muestra un resumen de todos los porcentajes de éxito en cada patrón

Protocolo	**Cantidad total**	**Validos**	**Porcentaje (%)**
TLS	28	20	71
DNS	94	94	100
DHCP 0	83	62	74
DHCP 152	179	179	100
HTTP	94	94	100
IGMP	241	241	100
MDNS	1213	1187	97,9
NTP	14	14	100
SSDP 1.1.1.2	304	232	76
SSDP 114	1092	1092	100

Tabla 2: Porcentaje de éxito de los patrones

4.9. Anomalía 1

Al filtrar por TCP se encontraron retransmisiones TLSv1.3 identificadas como [TCP Fast Retransmission] desde el servidor hacia el cliente. Este tipo de retransmisión ocurre cuando el servidor detecta rápidamente que ciertos segmentos no han sido reconocidos y decide reenviarlos sin esperar a que expire el temporizador de retransmisión estándar (RTO).

La presencia de numerosos eventos de este tipo puede deberse a ACKs retrasados o desordenados, congestión momentánea en la red, o inestabilidad en el enlace de comunicación. Aunque no necesariamente indican pérdida de paquetes permanente, sí reflejan que la conexión presenta un desempeño subóptimo, generando un uso poco eficiente del ancho de banda y posibles incrementos en la latencia percibida por las aplicaciones.

Filtro: `(tcp) && (_ws.col.info == "[TCP Fast Retransmission], Continuation Data")`

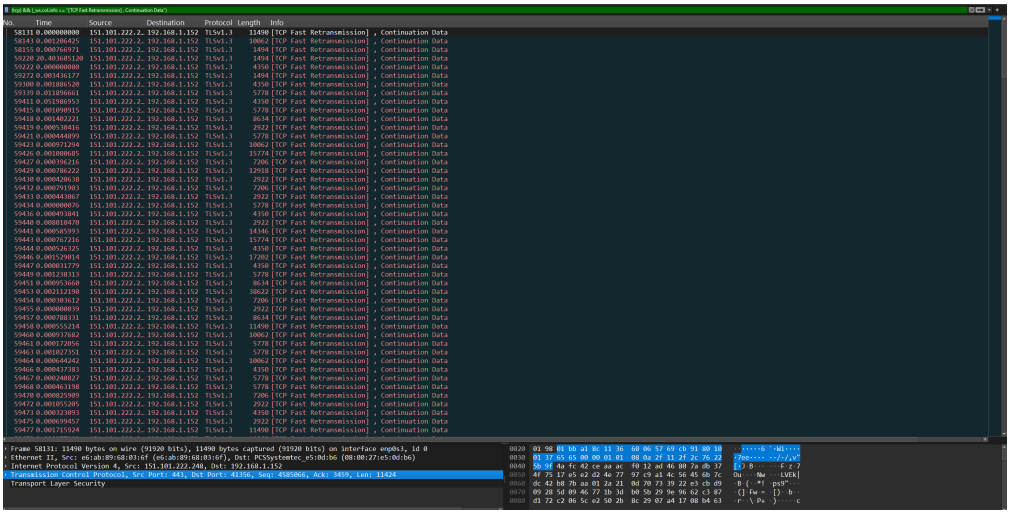

Figura 24: Captura anomalia 1: `TCP Fast Restransmission`

4.10. Anomalía 2

Al filtrar por TCP se encontraron se observaron múltiples eventos de tipo [TCP Dup ACK] enviados desde el cliente hacia el servidor. Los Duplicate ACKs ocurren cuando el receptor recibe segmentos fuera de orden y envía repetidamente el mismo número de confirmación (ACK) para indicar que espera aún un segmento perdido. Esta situación suele deberse a pérdida de paquetes en la red, variaciones en el orden de entrega o congestión temporal. La acumulación de estos mensajes es un indicio de que el flujo de datos está siendo interrumpido, forzando al emisor a reaccionar con retransmisiones rápidas, lo que puede degradar el rendimiento y aumentar la latencia percibida.

Filtro: `(tcp) && (tcp.analysis.duplicate_ack) && (ip.dst == 151.101.222.248)`

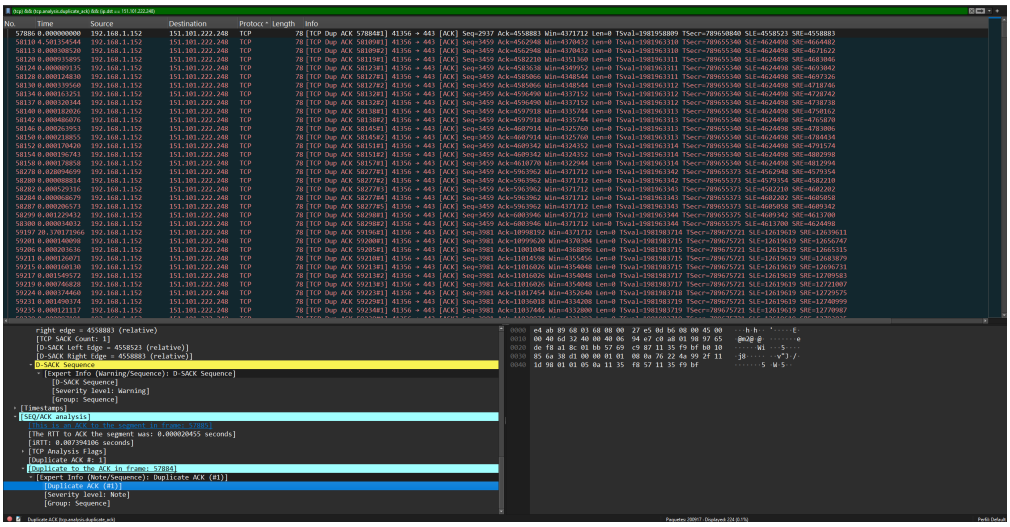

Figura 25: Captura anomalia 2: `TCP Duplicate ACK`

4.11. Anomalía 3

Al filtrar por TCP se identificaron eventos clasificados como [TCP Spurious Retransmission], donde el servidor retransmite segmentos que en realidad ya habían sido entregados correctamente. Este fenómeno ocurre típicamente cuando el emisor interpreta erróneamente la ausencia o retraso de ACKs como pérdida de paquetes, enviando datos redundantes. Las retransmisiones espurias no implican pérdida real, pero generan un uso ineficiente del ancho de banda y pueden introducir retrasos innecesarios en la sesión. Su presencia suele estar asociada a condiciones de jitter elevado, ACKs demorados o discrepancias temporales en la red.

Filtro: `(tcp) && (tcp.analysis.spurious_retransmission) && (ip.dst == 151.101.222.248)`

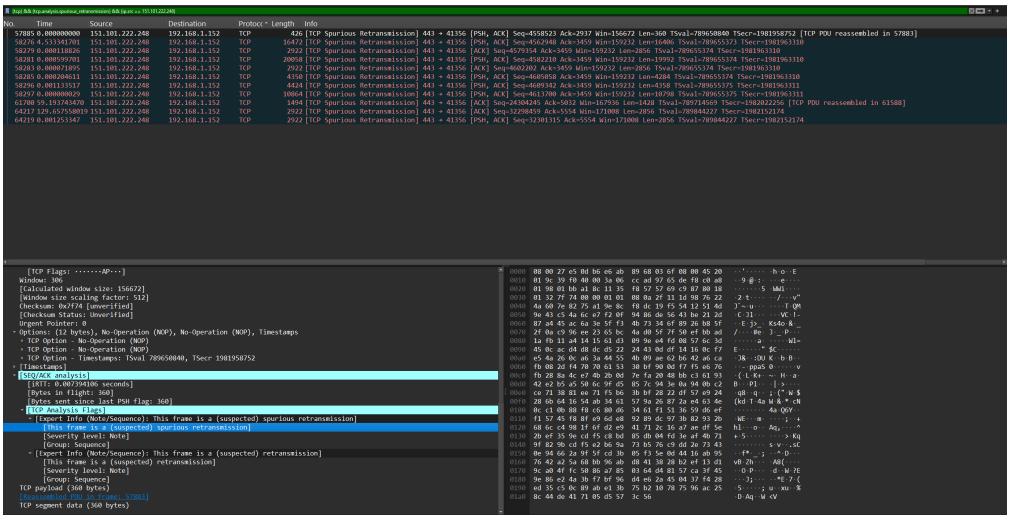

Figura 26: Captura anomalia 3: `TCP Spurious Retransmission`

4.12. Anomalía 4

Al filtrar por TCP se observaron múltiples segmentos identificados como [TCP Out-Of-Order]. Este tipo de eventos ocurre cuando los paquetes llegan al receptor en un orden distinto al esperado, lo cual obliga al cliente a reorganizar la secuencia antes de entregarla a la aplicación. Las causas más comunes de este comportamiento incluyen variaciones en las rutas de transmisión, latencias diferentes en enlaces intermedios o congestión temporal en la red. Aunque TCP está diseñado para manejar este tipo de situaciones mediante la reordenación de paquetes, una presencia constante de segmentos fuera de orden puede indicar problemas de estabilidad en la conexión, incrementando la latencia percibida y afectando la eficiencia de la transmisión.

Filtro: `(tcp) && (tcp.analysis.out_of_order) && (ip.dst == 151.101.222.248)`

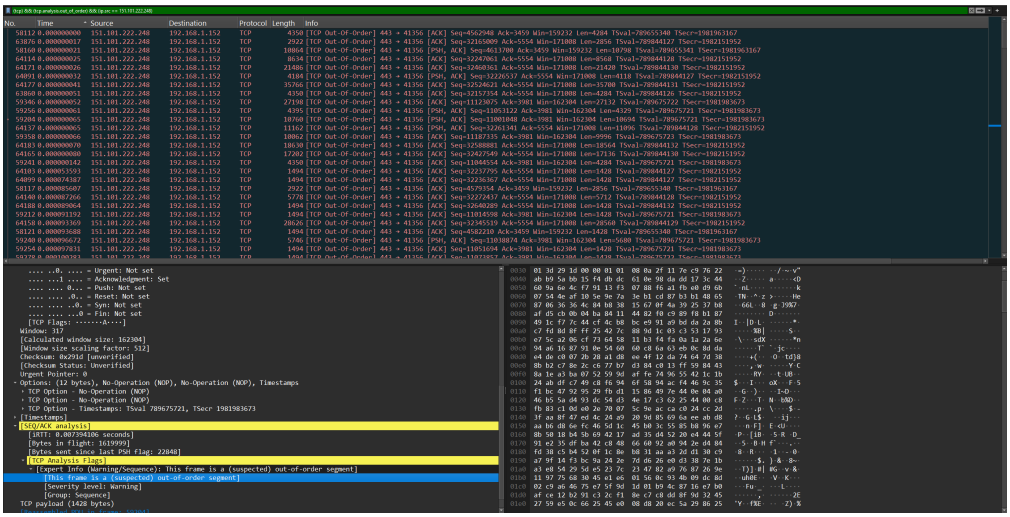

Figura 27: Captura anomalía 4: `TCP Out-Of-Order`

4.13. Anomalía 5

En la captura correspondiente se identificaron paquetes marcados con la bandera [RST], lo que indica que la conexión TCP fue terminada abruptamente mediante un reset de conexión. A diferencia del cierre normal con un intercambio de segmentos FIN-ACK, el envío de un RST fuerza la finalización inmediata de la sesión. Esto puede deberse a diversos factores, tales como una aplicación que decide abortar la comunicación, la intervención de un firewall, políticas de red que cortan la conexión o simplemente errores en la comunicación entre cliente y servidor. Este tipo de eventos son considerados anómalos, ya que afectan la continuidad de la sesión, pudiendo generar interrupciones visibles para el usuario final, como cortes en la transmisión de datos o fallas en el acceso a un servicio.

Filtro: `(_ws.expert) && (tcp.connection.rst) && (ip.dst == 151.101.222.248)`

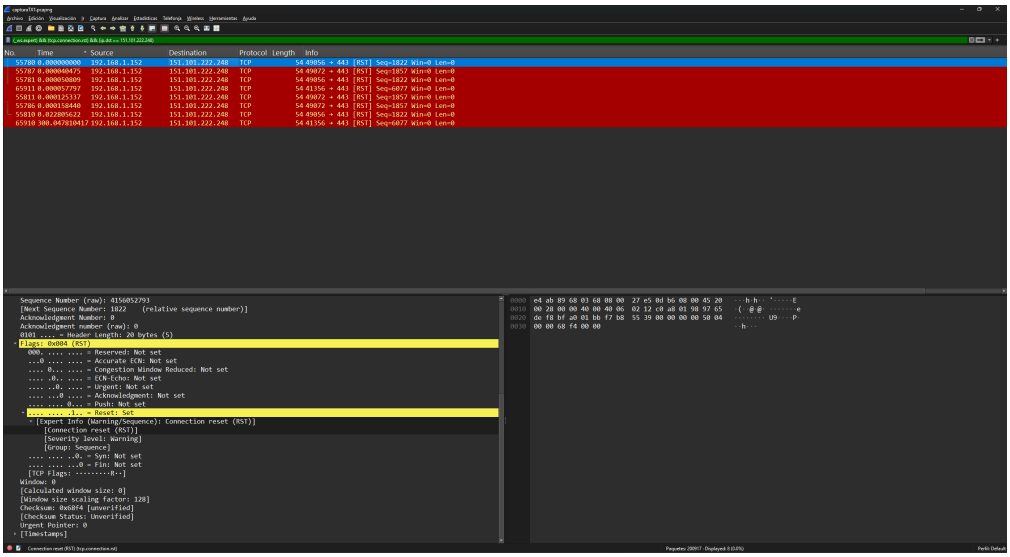

Figura 28: Captura anomalía 5: `TCP Reset (RST)`

5. Conclusión

El objetivo principal planteado en este trabajo consistía en obtener una captura extensa de tráfico de red y, a partir de ella, identificar al menos 10 patrones reconocibles aplicando filtros específicos en Wireshark, además de analizar dichos patrones para detectar posibles anomalías y explicar sus causas.

Se logró cumplir plenamente con este objetivo, ya que se realizó una captura con más de 200.000 paquetes registrados, lo que permitió disponer de bastante margen para encontrar patrones y anomalías. A partir de esta, se identificaron y documentaron 10 patrones distintos asociados a diferentes protocolos de red, entre ellos DNS, DHCP, HTTP, IGMP, MDNS, NTP, SSDP y TLS. En todos los casos, los patrones encontrados pudieron caracterizarse mediante gráficas que evidenciaron periodicidad, uniformidad o características propias del protocolo, lo que confirma la validez de los filtros aplicados.

Adicionalmente, se analizaron las posibles anomalías presentes, como saltos en la periodicidad, agrupaciones inesperadas de paquetes o retrasos en las respuestas, y se plantearon explicaciones técnicas coherentes respecto a estas irregularidades. La segunda captura enfocada en la aplicación YouTube permitió reforzar el aprendizaje al demostrar que los mismos criterios de filtrado y análisis pueden aplicarse en contextos más específicos, obteniendo también un patrón claro en las consultas DNS hacia el dominio principal de la plataforma.

Desde una perspectiva personal, el laboratorio permitió formar un juicio más claro sobre la importancia de contar con una base teórica sólida al momento de analizar tráfico de red. El uso de conceptos aprendidos en clases fue una de las principales fortalezas, ya que facilitó reconocer patrones y aplicar filtros de forma más sencilla. No obstante, también se hicieron evidentes ciertas debilidades, en particular la dificultad de identificar anomalías sin recurrir a varios conocimientos previos sobre el funcionamiento de los protocolos y su comportamiento esperado.

Como recomendación, considero que en futuros análisis es conveniente conocer de forma más precisa los protocolos o servicios que se buscan estudiar, lo cual permite enfocar mejor los filtros y reducir la complejidad del trabajo. En definitiva, este laboratorio no solo cumplió con lo propuesto en el enunciado, sino que también representó una experiencia enriquecedora al integrar la teoría con la práctica, reforzando competencias esenciales para diagnosticar, evaluar y comprender el tráfico de red en escenarios reales.

6. Bibliografía

- DACY, J. y HOWARTH, B. (1990). *Computer performance Management and Capacity Planning*. Prentice Hall, Sydney.
- CASAS, I. y AURTENECHEA, F. (1987). *Evaluación y Predicción del Desempeño de Sistemas Computacionales*. Actas XIII Conferencia Latinoamericana de Informática, Bogotá.
- UNISYS (1989). *U6000 Series System V, Administration Guide Vol 1*. Unisys Corporation.
- SEPULVEDA, M., NUSSBAUM, M., LAVAL, E. (1994). *A Shell for Approximation Methods*. 3rd. Pacific Rim International Conference on Artificial Intelligence, Beijing, China.