Redes de Comunicaciones I – Prácticas 2020
Práctica 3: Análisis de tráfico

Turno y pareja: 1392 - 5

Integrantes:

Fecha de entrega: 13/12/2020

Contenido

Contenido 1
1 Introducción 2
2 Realización de la práctica 2
3 Conclusiones 21

1 Introducción

La motivación de realizar esta práctica es que podamos analizar y entender el tráfico de una red para así poder administrarla correctamente. A continuación, realizaremos diversos análisis de los datos proporcionados por una traza de red de ejemplo para observar patrones en ella como sus clientes más demandantes, la carga de la red, y el posible uso que se le estaría dando a la red según los datos observados.

2 Realización de la práctica

1. Análisis de protocolos.

   Obtener los porcentajes de paquetes IP y NO IP (entendemos como **NO-IP** aquellos paquetes que no son ni **ETH|IP** ni **ETH|VLAN|IP**)

% Paquetes IP	% Paquetes NO-IP
91.88	8.12

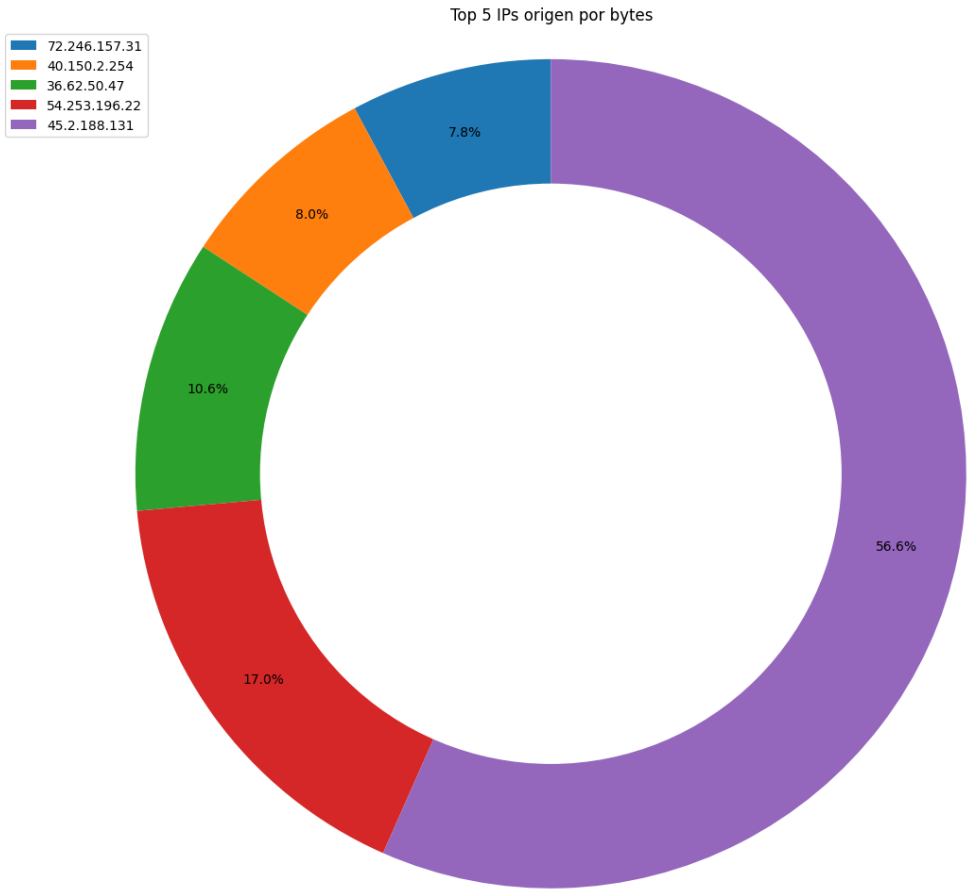
tshark -r <traza.pcap> -T fields -e eth.type -e ip.proto -Y 'eth.type eq 0x0800'
Contando los paquetes resultantes de ejecutar este comando y ya sabiendo el total de paquetes de la traza con el comando que ya venía de ejemplo, hemos obtenido los porcentajes.

   Obtener los porcentajes de paquetes UDP, TCP y OTROS sobre los que son IP (igualmente entienda, un paquete IP como aquel que cumpla la pila **ETH|IP** o **ETH|VLAN|IP**).

% Paquetes TCP	% Paquetes UDP	% Paquetes OTROS
62.62	1.50	35.88

Estos datos los hemos obtenido a la vez que obteníamos los de la tabla anterior, procesando la salida del comando anterior y comprobando si el ip.proto era 6 (TCP), 17 (UDP) u otros, aumentando el contador correspondiente en cada caso.

2. Obtención de top 5 de direcciones IP

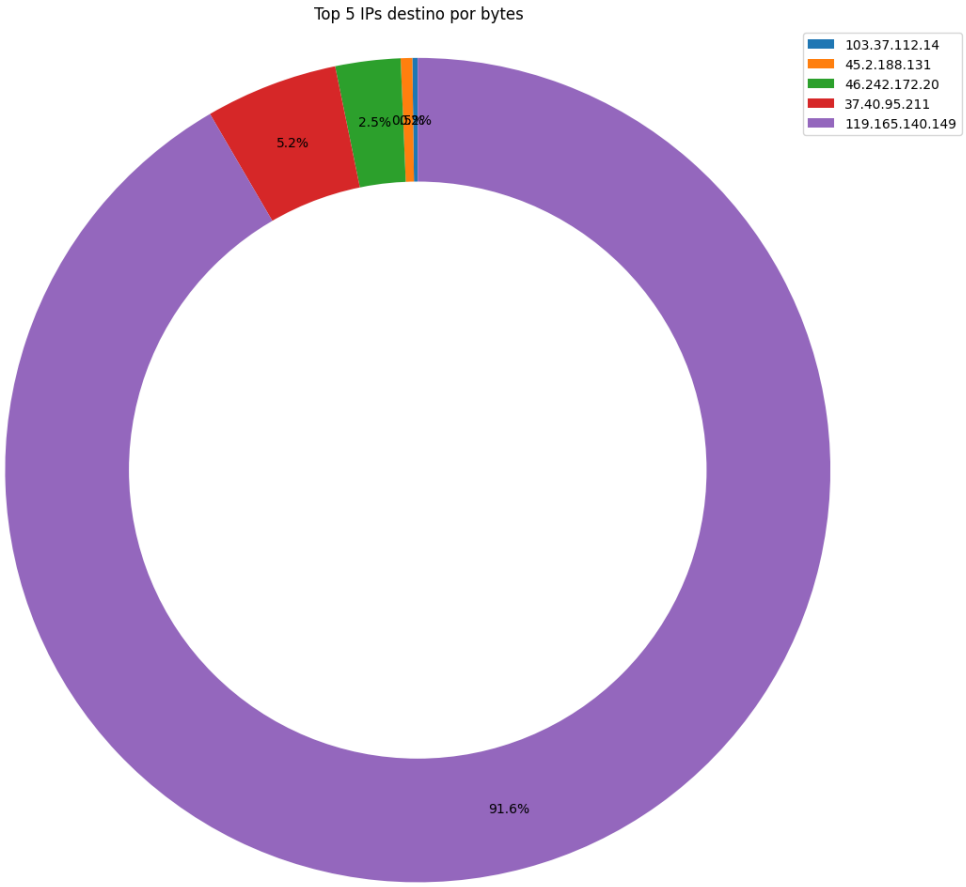


tshark -r <traza.pcap> -T fields -e ip.src -e frame.len -Y 'ip' | awk '{ips[$1] = ips[$1] + $2} END {for (ip in ips) {print ips[ip]"\t"ip}}' | sort -n -k 1 | tail -5

El único filtro tshark que hemos usado ha sido el ip, hemos hecho que tshark muestre dos columnas, una con ip.src y otra con frame.len. A continuación, hemos pasado esta salida por awk para, mediante un array hacer la suma acumulada de bytes por IP. Finalmente hemos ordenado el resultado en orden ascendente y hemos cogido las 5 últimas líneas, que son las correspondientes a las 5 IPs origen con más bytes, como se nos pedía.

Podemos observar como la IP de origen que más bytes ha mandado ha sido la 45.2.188.131 con un 56,6% del total de bytes enviado por las 5 IPs juntas. En orden decreciente podemos ver que la siguiente IP con más bytes enviados sería la 54.253.196.22 con 17% del total después la 36.62.50.47 con un 10,6% del total, a continuación, la 40.150.2.254 con un 8% del total y por último la 72.246.157.31 con un 7,8% del total de la suma de los bytes enviados por las 5 IPs.

Este tipo de análisis podría ser útil a la hora de determinar que usuario es el más demandante en cuanto a subida, también se podría detectar mediante este tipo de análisis, que usuario está realizando un ataque DOS ya que, seguramente, la IP de ese usuario destacaría como la número uno en este gráfico.

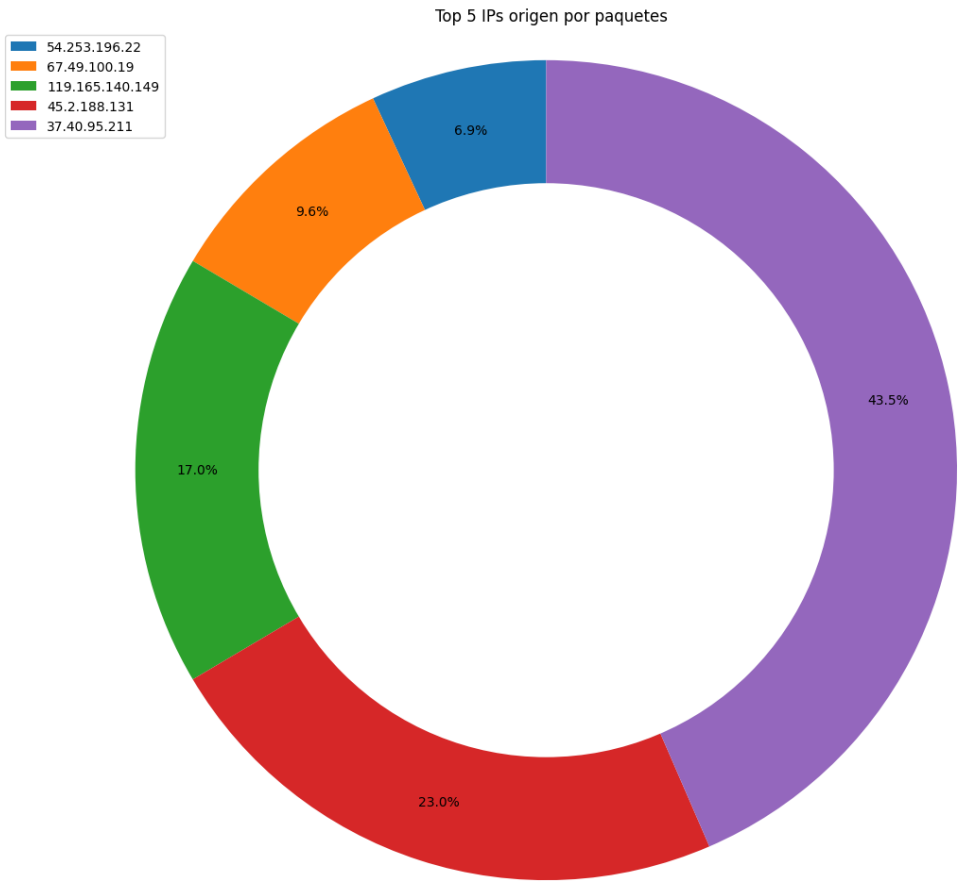


tshark -r <traza.pcap> -T fields -e ip.dst -e frame.len -Y 'ip' | awk '{ips[$1] = ips[$1] + $2} END {for (ip in ips) {print ips[ip]"\t"ip}}' | sort -n -k 1 | tail -5

El único filtro tshark que hemos usado ha sido el ip, hemos hecho que tshark muestre dos columnas, una con ip.dst y otra con frame.len. A continuación hemos pasado esta salida por awk para, mediante un array hacer la suma acumulada de bytes por IP. Finalmente hemos ordenado el resultado en orden ascendente y hemos cogido las 5 últimas líneas, que son las correspondientes a las 5 IPs destino con más bytes, como se nos pedía.

Podemos observar como la IP de destino que más bytes ha recibido ha sido la 119.165.140.149 con un 91,6% del total de bytes recibido por las 5 IPs juntas. En orden decreciente podemos ver que la siguiente IP con más bytes recibidos sería la 37.40.95.211 con 5,2% del total después la 46.242.172.20 con un 2,5% del total, a continuación la 45.2.188.131 con un 0,5% del total y por último la 103.37.112.14 con un 0,2% del total de la suma de los bytes recibidos por las 5 IPs.

Este tipo de análisis podría ser útil a la hora de determinar que usuario es el más demandante en cuanto a bajada, también se podría detectar mediante este tipo de análisis, que usuario (si este fuera algún tipo de servidor) está siendo víctima de un ataque DOS ya que, seguramente, la IP de ese usuario destacaría como la número uno en este gráfico.

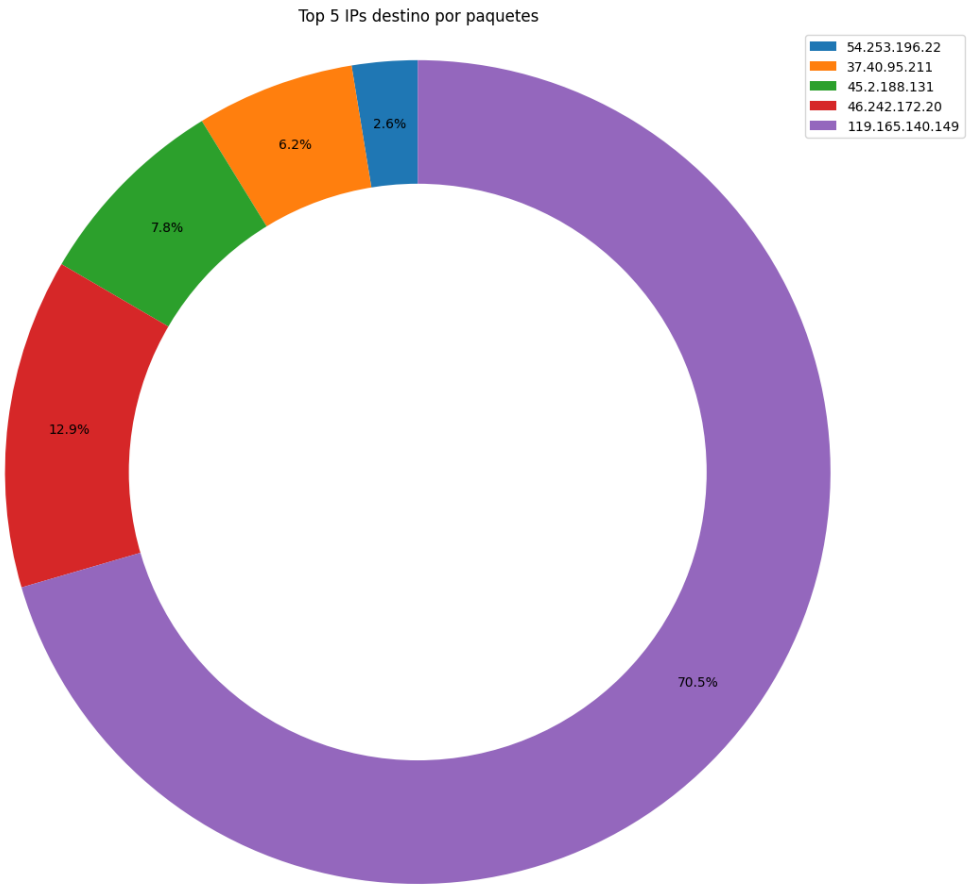


tshark -r <traza.pcap> -T fields -e ip.src -Y 'ip' | awk '{ips[$1] = ips[$1] + 1} END {for (ip in ips) {print ips[ip]"\t"ip}}' | sort -n -k 1 | tail -5

El único filtro tshark que hemos usado ha sido el ip, hemos hecho que tshark muestre una columna con ip.src. A continuación hemos pasado esta salida por awk para, mediante un array contar los paquetes por IP. Finalmente hemos ordenado el resultado en orden ascendente y hemos cogido las 5 últimas líneas, que son las correspondientes a las 5 IPs origen con más paquetes, como se nos pedía.

Podemos observar como la IP de origen que más paquetes ha enviado ha sido la 37.40.95.211 con un 43,5% del total de paquetes enviados por las 5 IPs juntas. En orden decreciente podemos ver que la siguiente IP con más paquetes enviados sería la 45.2.188.131 con 23% del total después la 119.165.140.149 con un 17% del total, a continuación la 67.49.100.19 con un 9,6% del total y por último la 54.253.196.22 con un 6,9% del total de la suma de los paquetes enviados por las 5 IPs.

Este tipo de análisis podría ser útil a la hora de determinar que usuario es el más demandante en cuanto a subida, también se podría detectar mediante este tipo de análisis y su homólogo de bytes, que usuario está realizando un ataque DOS ya que, seguramente, la IP de ese usuario destacaría como la número uno en ambos gráficos.



tshark -r <traza.pcap> -T fields -e ip.dst -Y 'ip' | awk '{ips[$1] = ips[$1] + 1} END {for (ip in ips) {print ips[ip]"\t"ip}}' | sort -n -k 1 | tail -5

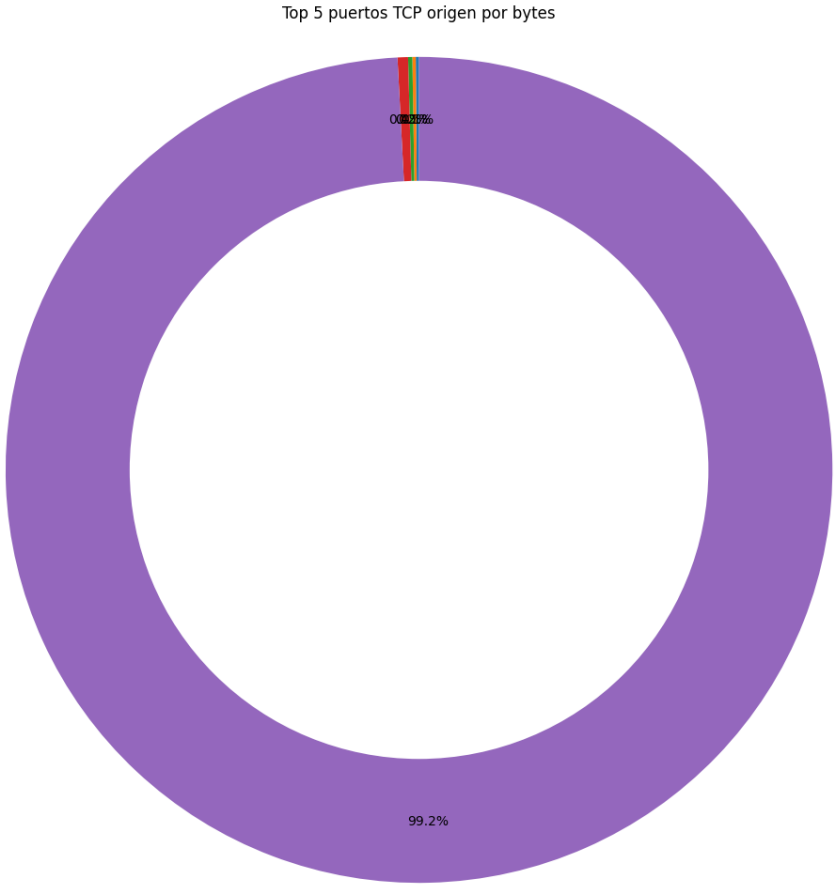
El único filtro tshark que hemos usado ha sido el ip, hemos hecho que tshark muestre una columna con ip.dst. A continuación hemos pasado esta salida por awk para, mediante un array contar los paquetes por IP. Finalmente hemos ordenado el resultado en orden ascendente y hemos cogido las 5 últimas líneas, que son las correspondientes a las 5 IPs destino con más paquetes, como se nos pedía.

Podemos observar como la IP de destino que más paquetes ha recibido ha sido la 119.165.140.149 con un 70,5% del total de paquetes recibido por las 5 IPs juntas. En orden decreciente podemos ver que la siguiente IP con más paquetes recibidos sería la 46.242.172.20 con 12,9% del total después la 45.2.188.131 con un 7,8% del total, a continuación la 37.40.95.211 con un 6,2% del total y por último la 54.253.196.22 con un 2.6% del total de la suma de los paquetes recibidos por las 5 IPs.

Este tipo de análisis podría ser útil a la hora de determinar que usuario es el más demandante en cuanto a bajada, también se podría detectar mediante este tipo de análisis y su homólogo de bytes, que usuario (si este fuera algún tipo de servidor) está siendo víctima de un ataque DOS ya que, seguramente, la IP de ese usuario destacaría como el número uno en ambos gráficos.

3. Obtención de top 5 de puertos:

TCP:

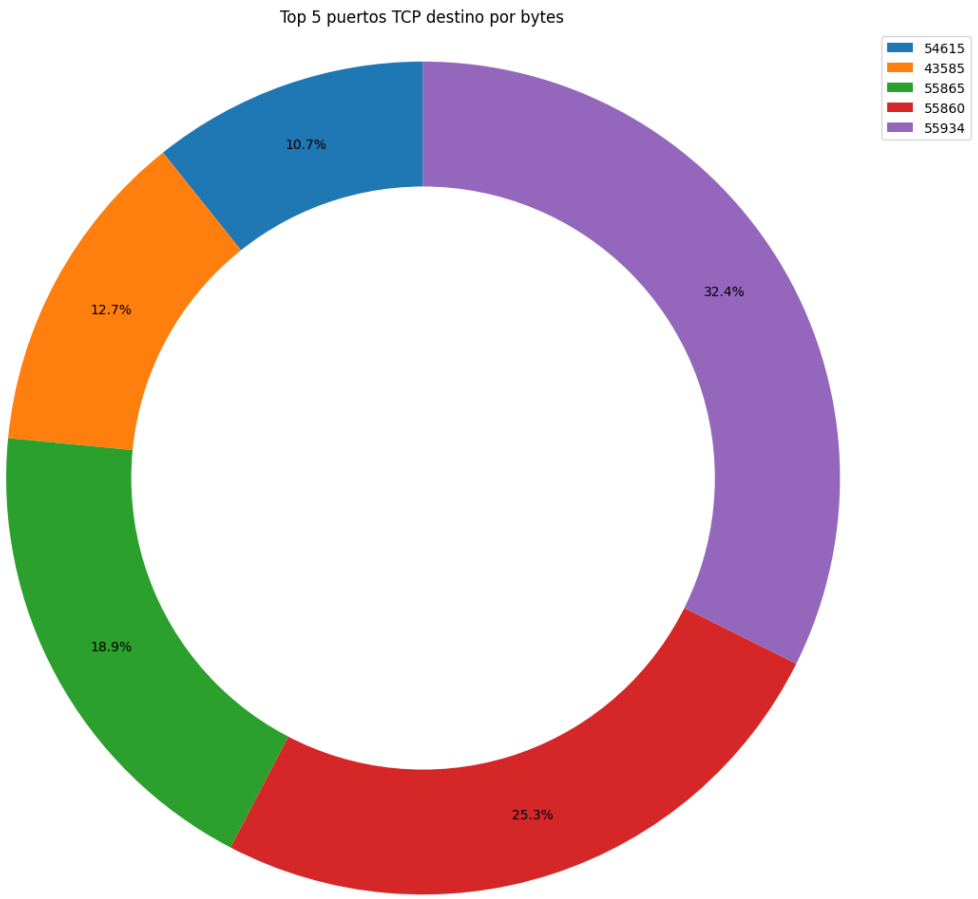


tshark -r <traza.pcap> -T fields -e tcp.srcport -e frame.len -Y 'tcp' | awk '{ips[$1] = ips[$1] + $2} END {for (ip in ips) {print ips[ip]"\t"ip}}' | sort -n -k 1 | tail -5

El único filtro tshark que hemos usado ha sido el tcp, hemos hecho que tshark muestre dos columnas, una con tcp.srcport y la otra con frame.len. A continuación, hemos pasado esta salida por awk para, mediante un array hacer la suma acumulada de bytes por puerto. Finalmente hemos ordenado el resultado en orden ascendente y hemos cogido las 5 últimas líneas, que son las correspondientes a los 5 puertos origen con más bytes, como se nos pedía.

Podemos observar como el puerto origen que más bytes ha mandado ha sido el 80 con un 99,2% del total de bytes enviado por los 5 puertos juntos. El 0,8% restante, en orden decreciente se lo reparten los puertos 443, 55934, 54615, 55860.

Como hemos visto el puerto que más bytes ha enviado ha sido el 80, esto podría indicarnos que el sistema cuenta con un servidor de una página web.

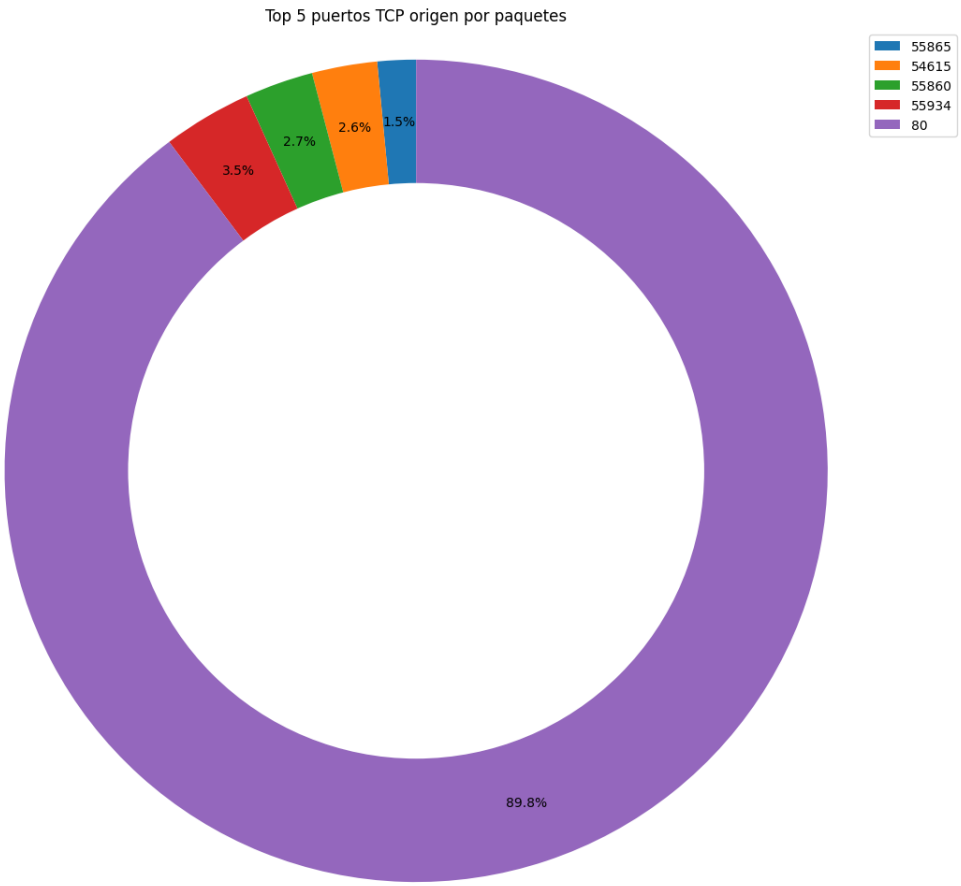


tshark -r <traza.pcap> -T fields -e tcp.dstport -e frame.len -Y 'tcp' | awk '{ips[$1] = ips[$1] + $2} END {for (ip in ips) {print ips[ip]"\t"ip}}' | sort -n -k 1 | tail -5

El único filtro tshark que hemos usado ha sido el tcp, hemos hecho que tshark muestre dos columnas, una con tcp.dstport y la otra con frame.len. A continuación, hemos pasado esta salida por awk para, mediante un array hacer la suma acumulada de bytes por puerto. Finalmente hemos ordenado el resultado en orden ascendente y hemos cogido las 5 últimas líneas, que son las correspondientes a los 5 puertos destino con más bytes, como se nos pedía.

Podemos observar como el puerto destino que más bytes ha recibido ha sido el 55934 con un 32,4% del total de bytes recibido por los 5 puertos juntos. En orden decreciente podemos ver que el siguiente puerto con más bytes recibidos sería el 55860 con 25,3% del total después el 55865 con un 18,9% del total, a continuación el 43585 con un 12,7% del total y por último el 54615 con un 10,7% del total de la suma de los bytes recibidos por los 5 puertos.

Como hemos descrito, el puerto con más tráfico de entrada es el 55934, según lo que hemos podido ver en internet, este puerto podría estar asociado a un servidor proxy.

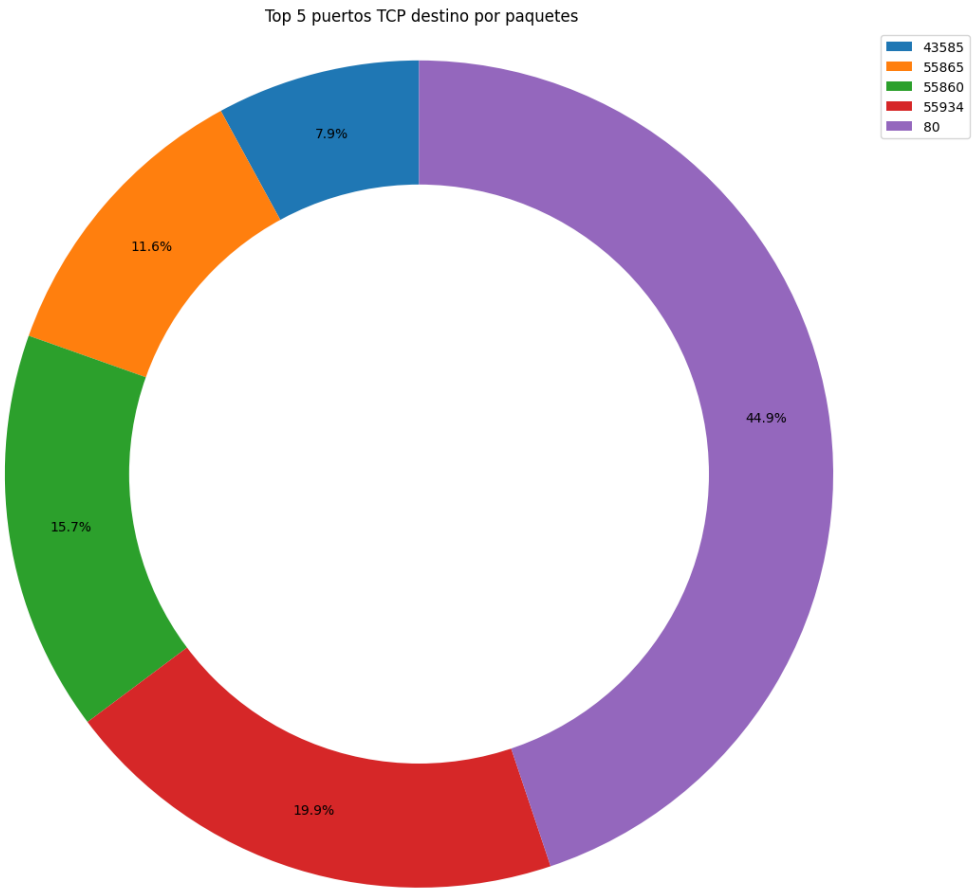


tshark -r <traza.pcap> -T fields -e tcp.srcport -Y 'tcp' | awk '{ips[$1] = ips[$1] + 1} END {for (ip in ips) {print ips[ip]"\t"ip}}' | sort -n -k 1 | tail -5

El único filtro tshark que hemos usado ha sido el tcp, hemos hecho que tshark muestre una columna con tcp.srcport. A continuación, hemos pasado esta salida por awk para, mediante un array hacer la suma de los paquetes por puerto. Finalmente hemos ordenado el resultado en orden ascendente y hemos cogido las 5 últimas líneas, que son las correspondientes a los 5 puertos origen con más paquetes, como se nos pedía.

Podemos observar como el puerto origen que más paquetes ha mandado ha sido el 80 con un 89,8% del total de paquetes enviado por los 5 puertos juntos. En orden decreciente podemos ver que el siguiente puerto que más paquetes ha enviados sería el 55934 con 3,5% del total después el 55860 con un 2,7% del total, a continuación, el 54615 con un 2,6% del total y por último el 55865 con un 1,5% del total de la suma de los paquetes enviados por los 5 puertos.

Como hemos visto el puerto que más paquetes ha enviado ha sido el 80, esto refuerza lo dicho antes sobre que el sistema se podría tratar de un servidor web.



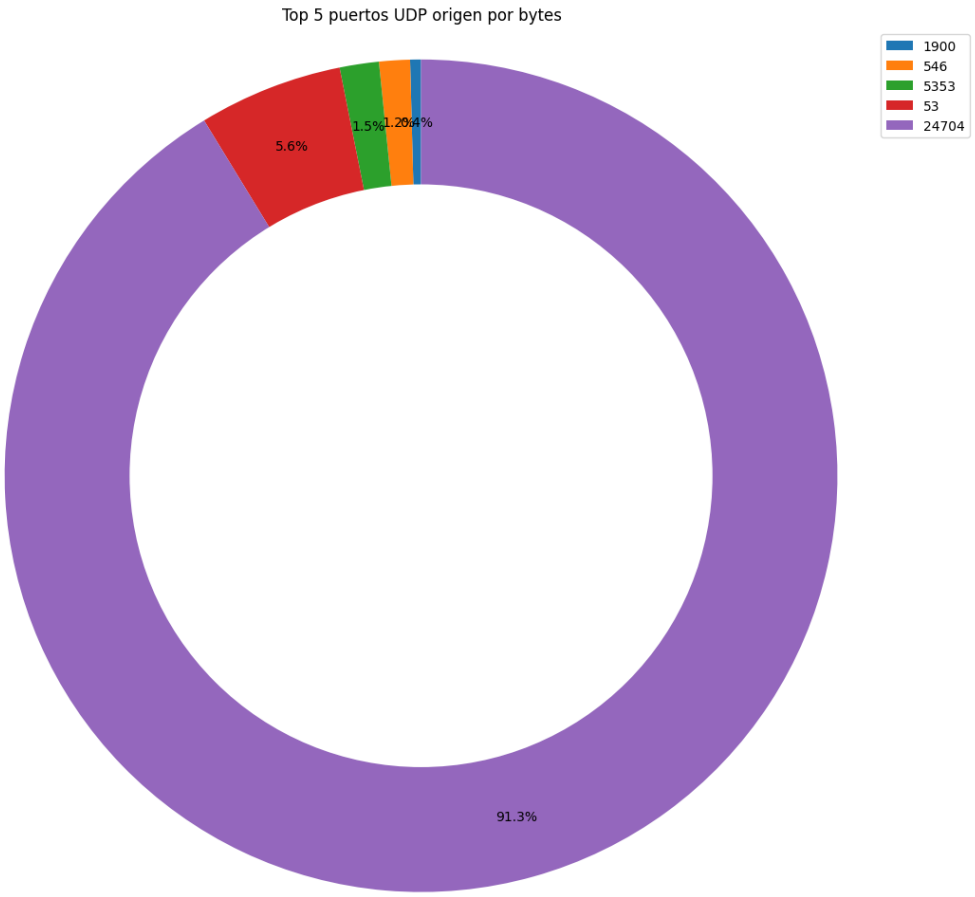
tshark -r <traza.pcap> -T fields -e tcp.dstport -Y 'tcp' | awk '{ips[$1] = ips[$1] + 1} END {for (ip in ips) {print ips[ip]"\t"ip}}' | sort -n -k 1 | tail -5

El único filtro tshark que hemos usado ha sido el tcp, hemos hecho que tshark muestre una columna con tcp.dstport. A continuación, hemos pasado esta salida por awk para, mediante un array hacer la suma de los paquetes por puerto. Finalmente hemos ordenado el resultado en orden ascendente y hemos cogido las 5 últimas líneas, que son las correspondientes a los 5 puertos destino con más paquetes, como se nos pedía.

Podemos observar como el puerto destino que más paquetes ha recibido ha sido el 80 con un 44,9% del total de paquetes recibidos por los 5 puertos juntos. En orden decreciente podemos ver que el siguiente puerto que más paquetes ha recibido sería el 55934 con 19,9% del total después el 55860 con un 15,7% del total, a continuación, el 55865 con un 11,6% del total y por último el 43585 con un 7,9% del total de la suma de los paquetes recibidos por los 5 puertos.

Como hemos visto el puerto que más paquetes ha recibido ha sido el 80, esto podría indicarnos que muchas de las máquinas que se encuentran en la red están haciendo peticiones a un servidor de una página web, es decir, navegando por internet.

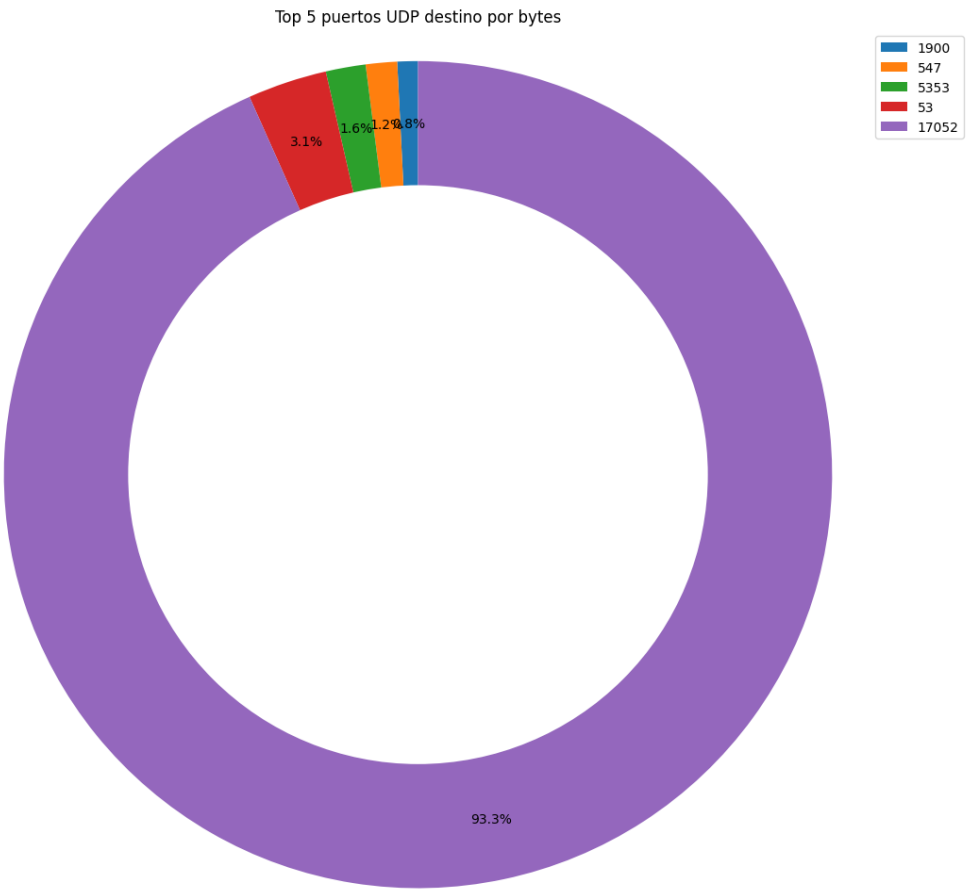
UDP:



tshark -r <traza.pcap> -T fields -e udp.srcport -e frame.len -Y 'udp' | awk '{ips[$1] = ips[$1] + $2} END {for (ip in ips) {print ips[ip]"\t"ip}}' | sort -n -k 1 | tail -5

El único filtro tshark que hemos usado ha sido el udp, hemos hecho que tshark muestre dos columnas, una con udp.srcport y la otra con frame.len. A continuación, hemos pasado esta salida por awk para, mediante un array hacer la suma acumulada de bytes por puerto. Finalmente hemos ordenado el resultado en orden ascendente y hemos cogido las 5 últimas líneas, que son las correspondientes a los 5 puertos origen con más bytes, como se nos pedía.

Podemos observar como el puerto origen que más bytes ha mandado ha sido el 24704 con un 91,3% del total de bytes enviado por los 5 puertos juntos. En orden decreciente podemos ver que el siguiente puerto con más bytes enviados sería el 53 con 5,6% del total después el 5353 con un 1,5% del total, a continuación el 546 con un 1,2% del total y por último el 1900 con un 0,4% del total de la suma de los bytes enviados por los 5 puertos.
Como hemos visto el puerto que más bytes ha enviado ha sido el 24704, esto podría indicarnos que el sistema está retransmitiendo por ese puerto un flujo de información constante que no necesariamente tiene que llegar a su destino, como podría ser una videollamada.

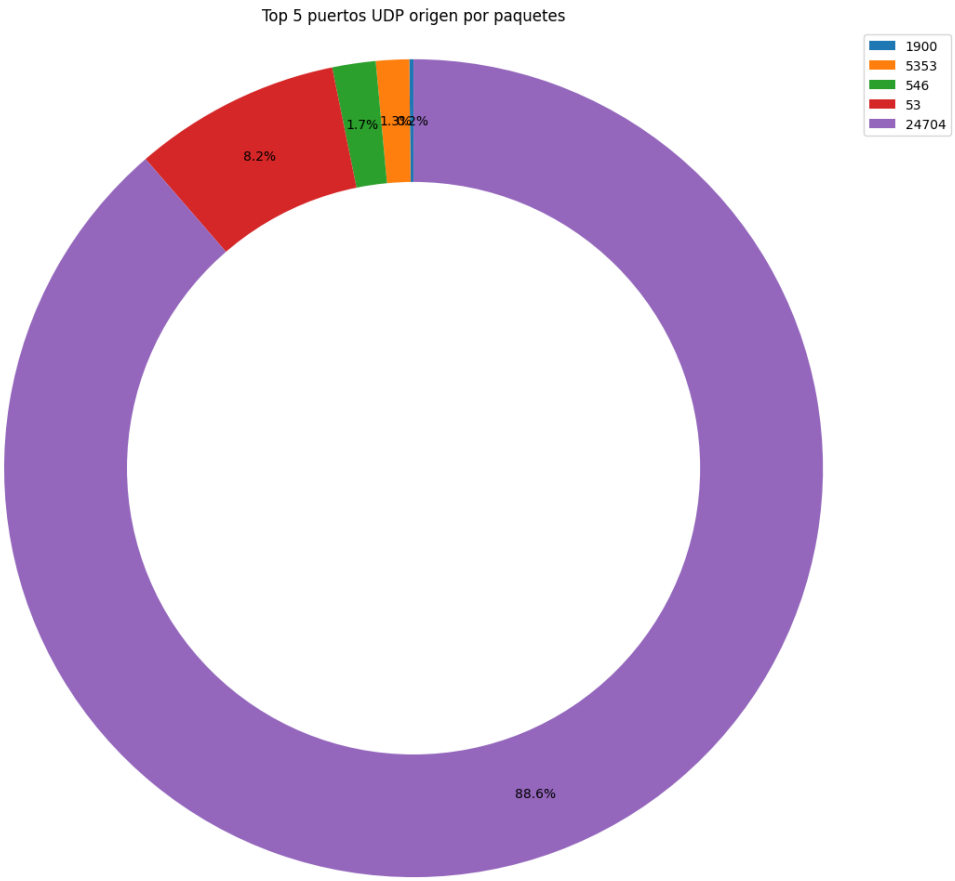


tshark -r <traza.pcap> -T fields -e udp.dstport -e frame.len -Y 'udp' | awk '{ips[$1] = ips[$1] + $2} END {for (ip in ips) {print ips[ip]"\t"ip}}' | sort -n -k 1 | tail -5

El único filtro tshark que hemos usado ha sido el udp, hemos hecho que tshark muestre dos columnas, una con udp.dstport y la otra con frame.len. A continuación, hemos pasado esta salida por awk para, mediante un array hacer la suma acumulada de bytes por puerto. Finalmente hemos ordenado el resultado en orden ascendente y hemos cogido las 5 últimas líneas, que son las correspondientes a los 5 puertos destino con más bytes, como se nos pedía.

Podemos observar como el puerto destino que más bytes ha recibido ha sido el 17052 con un 93,3% del total de bytes recibido por los 5 puertos juntos. En orden decreciente podemos ver que el siguiente puerto con más bytes recibidos sería el 53 con 3,1% del total después el 5353 con un 1,6% del total, a continuación el 547 con un 1,2% del total y por último el 1900 con un 0,8% del total de la suma de los bytes recibidos por los 5 puertos.

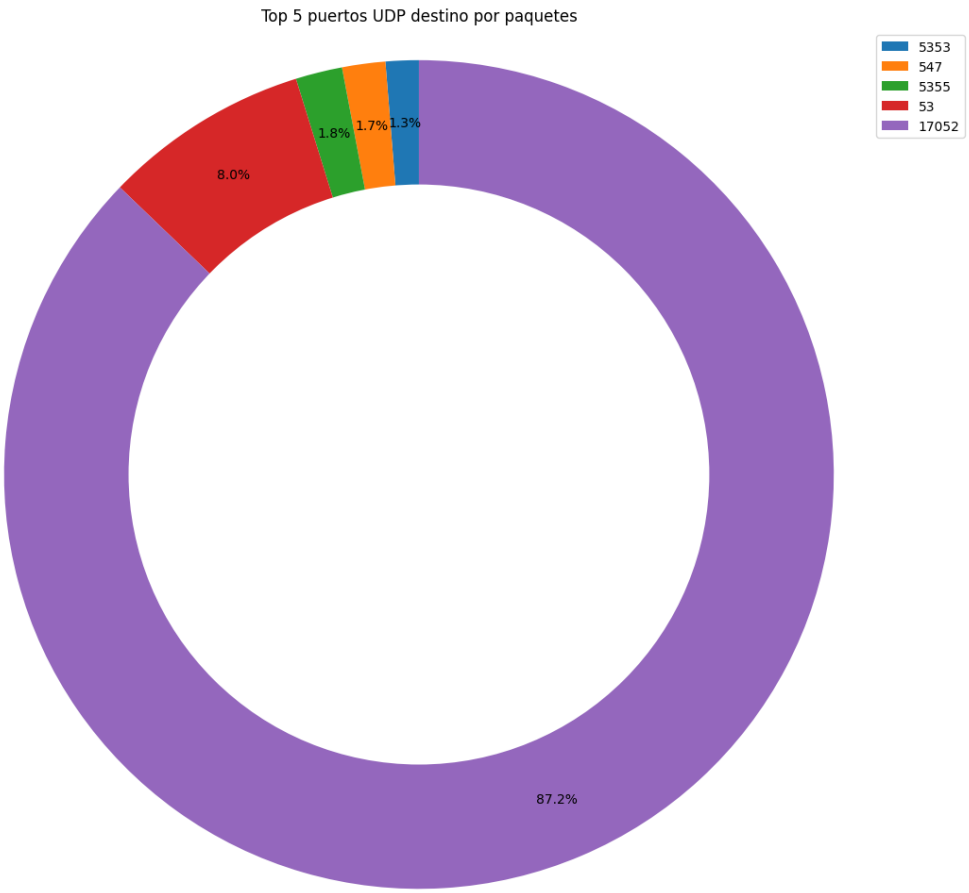
Como hemos descrito, el puerto con más tráfico de entrada es el 17052, esto podría indicarnos que el sistema está recibiendo un flujo de información constante que no necesariamente le tiene que llegar correctamente, como podría ser una videollamada.



tshark -r <traza.pcap> -T fields -e udp.srcport -Y 'udp' | awk '{ips[$1] = ips[$1] + 1} END {for (ip in ips) {print ips[ip]"\t"ip}}' | sort -n -k 1 | tail -5
El único filtro tshark que hemos usado ha sido el udp, hemos hecho que tshark muestre una columna con udp.srcport. A continuación, hemos pasado esta salida por awk para, mediante un array hacer la suma de los paquetes por puerto. Finalmente hemos ordenado el resultado en orden ascendente y hemos cogido las 5 últimas líneas, que son las correspondientes a los 5 puertos origen con más paquetes, como se nos pedía.

Podemos observar como el puerto origen que más paquetes ha mandado ha sido el 24704 con un 88,6% del total de paquetes enviado por los 5 puertos juntos. En orden decreciente podemos ver que el siguiente puerto que más paquetes ha enviados sería el 53 con 8,2% del total después el 546 con un 1,7% del total, a continuación, el 5353 con un 1,3% del total y por último el 1900 con un 0,2% del total de la suma de los paquetes enviados por los 5 puertos.

Como hemos visto el puerto que más paquetes ha enviado ha sido el 24704, esto refuerza lo dicho antes sobre que el sistema podría estar realizando algún tipo de servicio de streaming, como una videollamada, que necesita mucho ancho de banda para poder realizarse correctamente.



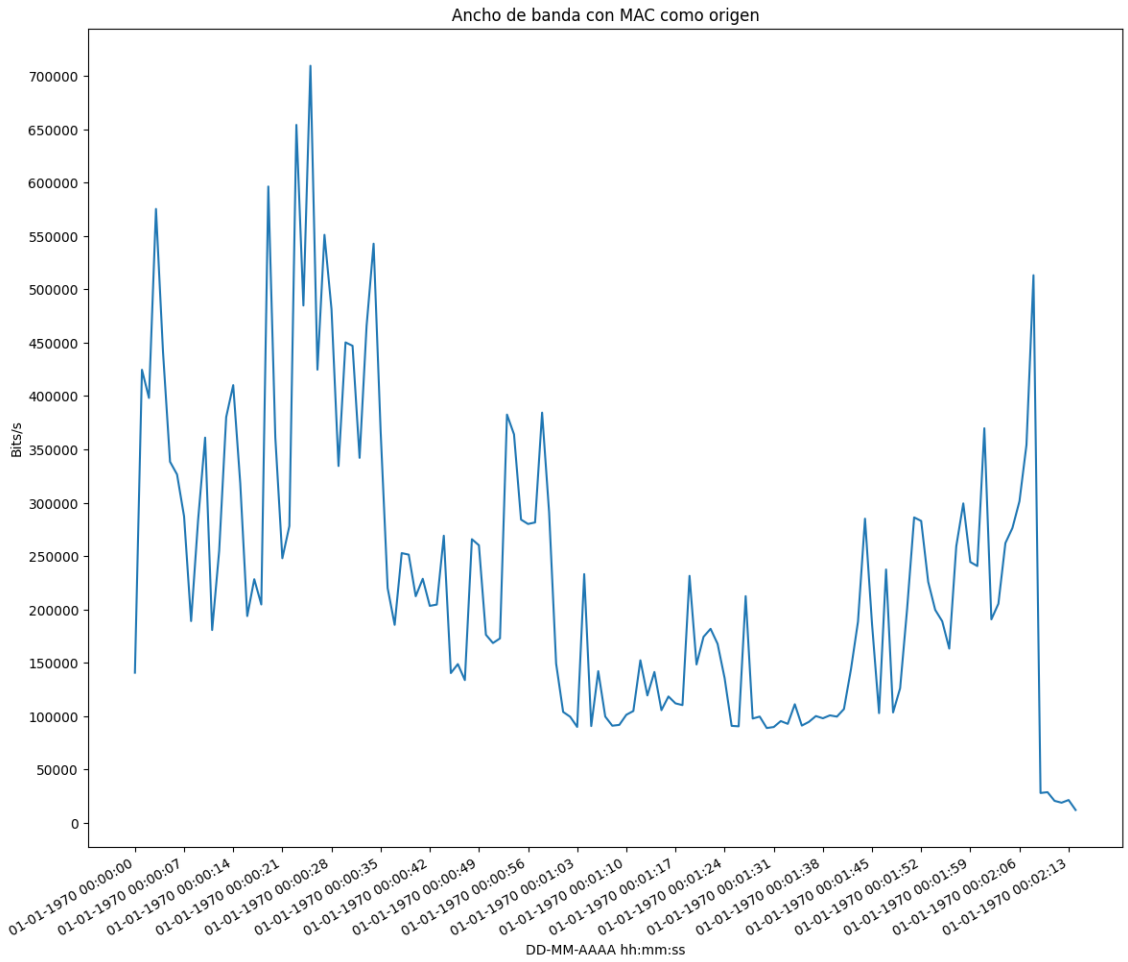
tshark -r <traza.pcap> -T fields -e udp.dstport -Y 'tcp' | awk '{ips[$1] = ips[$1] + 1} END {for (ip in ips) {print ips[ip]"\t"ip}}' | sort -n -k 1 | tail -5
El único filtro tshark que hemos usado ha sido el udp, hemos hecho que tshark muestre una columna con udp.dstport. A continuación, hemos pasado esta salida por awk para, mediante un array hacer la suma de los paquetes por puerto. Finalmente hemos ordenado el resultado en orden ascendente y hemos cogido las 5 últimas líneas, que son las correspondientes a los 5 puertos destino con más paquetes, como se nos pedía.

Podemos observar como el puerto destino que más paquetes ha recibido ha sido el 17052 con un 87,2% del total de paquetes recibidos por los 5 puertos juntos. En orden decreciente podemos ver que el siguiente puerto que más paquetes ha recibido sería el 53 con 8% del total después el 5355 con un 1,8% del total, a continuación, el 547 con un 1,7% del total y por último el 5353 con un 1,3% del total de la suma de los paquetes recibidos por los 5 puertos.

Como hemos visto el puerto que más paquetes ha recibido ha sido el 17052, esto refuerza lo dicho antes sobre que el sistema podría estar recibiendo algún tipo de servicio de streaming, como una videollamada, que necesita mucho ancho de banda para poder realizarse correctamente.

4. Series temporales de ancho de banda/tasa/caudal:

La dirección MAC usada para filtrar las series temporales y separar los sentidos es 00:11:88:CC:33:5A.



Para obtener los datos de la serie temporal del ancho de banda cuando la MAC usada es el origen de la comunicación, hemos utilizado el siguiente filtro:

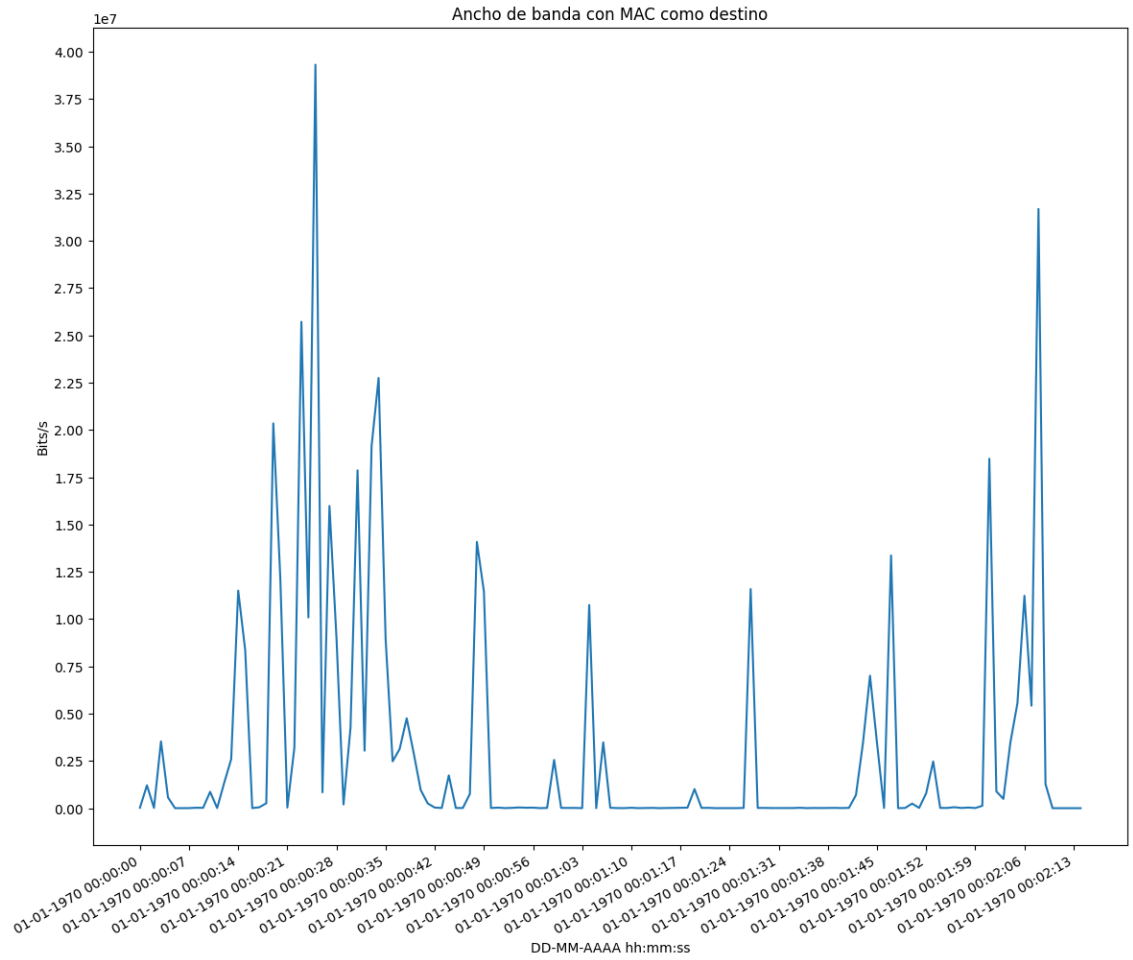
tshark -r <traza.pcap> -qz io,stat,1,,eth.src==00:11:88:CC:33:5A | grep "<>" | awk '{print $8*8}'

A través de la opción "-qz io,stat,1" expresamos que queremos obtener una serie con granularidad de 1 segundo, a continuación, con "eth.src==00:11:88:CC:33:5A" indicamos que queremos filtrar los paquetes que tienen esa dirección MAC origen.

Finalmente, como el resultado de la primera parte del comando nos devuelve una tabla, simplemente seleccionamos aquellas líneas que contienen "<>", puesto que es donde se encuentran los datos. A continuación, la columna que nos interesa es la de "Bytes", es decir la columna 8, y para pasarla a bits la multiplicamos por 8 (awk '{print $8*8}').

En la gráfica podemos observar como el ancho de banda que se consume cuando se envían paquetes desde la dirección MAC en cuestión no es relativamente grande (como veremos en la próxima gráfica). Por tanto, podemos decir que el dispositivo al que corresponde esta dirección MAC no envía paquetes de gran tamaño (los cuales consumen un mayor ancho de banda).

Además, podemos apreciar una gran cantidad de picos y valles, estos muestran los momentos en los que el dispositivo envía un número mayor o menor de bits por segundo, es decir, mayor o menor información respectivamente.



Para obtener los datos de la serie temporal del ancho de banda cuando la MAC usada es el destino de la comunicación, hemos utilizado el siguiente filtro:

tshark -r <traza.pcap> -qz io,stat,1,,eth.dst==00:11:88:CC:33:5A | grep "<>" | awk '{print $8*8}'

Equivalente al filtro usado anteriormente para cuando la MAC era usada como destino, simplemente cambiando “eth.src” por “eth.dst”. De todas formas, a continuación lo explicaremos en detalle.

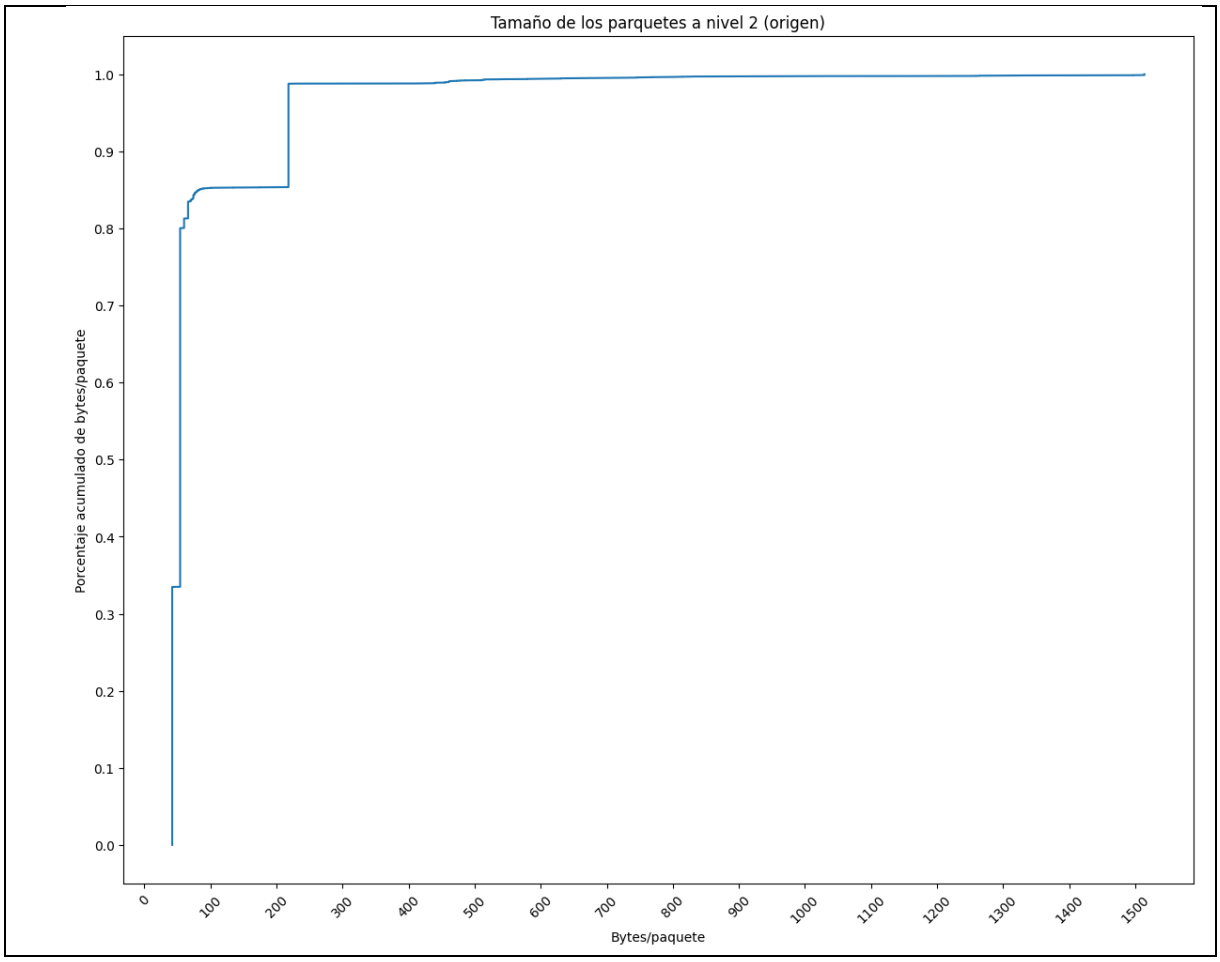
A través de la opción “-qz io,stat,1” expresamos que queremos obtener una serie con granularidad de 1 segundo, luego, con “eth.dst==00:11:88:CC:33:5A” indicamos que queremos filtrar los paquetes que tienen esa dirección MAC destino.

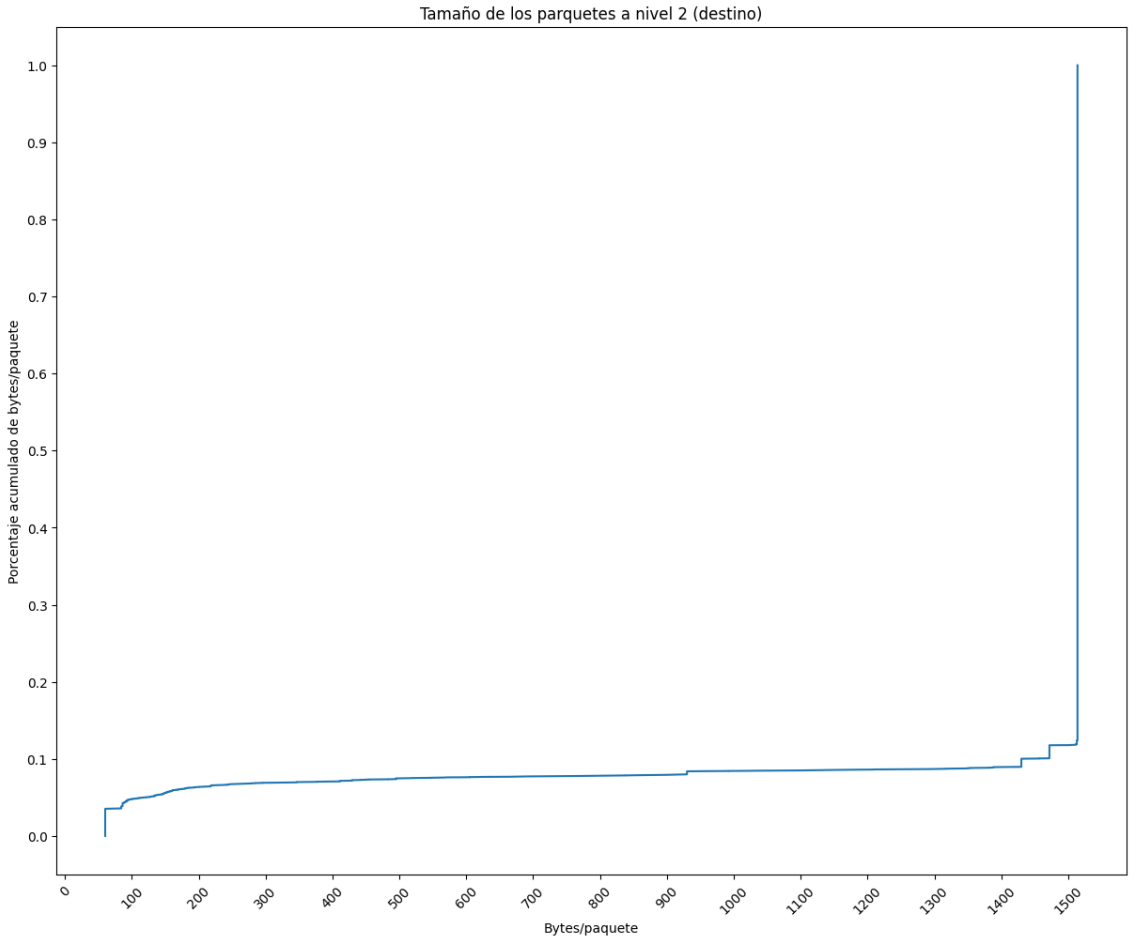
Finalmente, como el resultado de la primera parte del comando nos devuelve una tabla, simplemente seleccionamos aquellas líneas que contienen “<>”, puesto que es donde se encuentran los datos. A continuación, la columna que nos interesa es la de “Bytes”, es decir la columna 8, y para pasarla a bits la multiplicamos por 8 (awk '{print $8*8}').

En la gráfica podemos observar valores bastante más grandes que en la gráfica de la MAC como origen, incluso aquí tenemos momentos en los que no existe tráfico en sentido destino (representado como 0). Debemos darnos cuenta de que en los momentos que este dispositivo recibe más información coincide también, generalmente, en los momentos en los que, a su vez, ha enviado más información.

Por todo ello, podríamos pensar que el dispositivo puede ser un ordenador, por ejemplo, que realiza consultas en una web, las cuales no consumen gran cantidad de ancho de banda, pero la información que recibe de esas consultas sí, como puede ser la descarga de archivos. Además, los momentos en los que se ha enviado información y no existe tráfico en sentido destino, podemos interpretarlo como que no existe una respuesta.

5. ECDFs de los tamaños de los paquetes





Para obtener los datos de las gráficas del ECDF del tamaño de todos los paquetes (a nivel 2), hemos utilizado los siguientes filtros:

- Sentido origen: **tshark -r <traza.pcap> -T fields -e frame.len -Y 'eth.src eq 00:11:88:CC:33:5A'**
- Sentido destino: **tshark -r <traza.pcap> -T fields -e frame.len -Y 'eth.dst eq 00:11:88:CC:33:5A'**

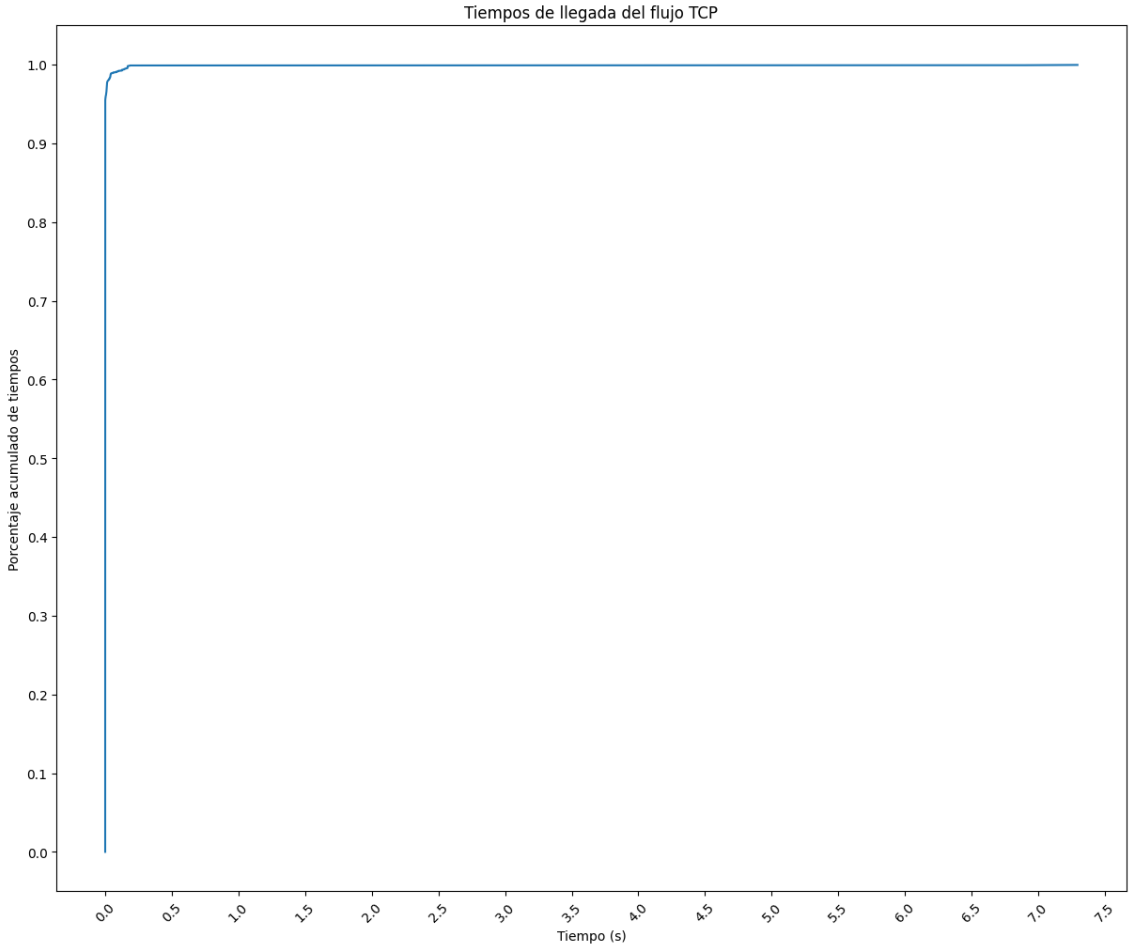
Primero filtramos aquellos paquetes cuya dirección MAC origen o destino, según corresponda, es 00:11:88:CC:33:5A, y a continuación, obtenemos el tamaño en bytes de cada uno de estos paquetes.

Si nos fijamos en la gráfica de sentido origen, podemos apreciar como la mayor parte de los paquetes que envía el dispositivo con esta dirección MAC tienen un tamaño entre 50 y 100 bytes aproximadamente. Luego, una ínfima parte de ellos tienen un tamaño aproximado de 200 bytes. Por tanto, podemos decir que este dispositivo no envía paquetes de gran tamaño.

Por el contrario, si nos fijamos en la gráfica de sentido destino, observamos como la mayor parte de los paquetes que recibe el dispositivo son de unos 1500 bytes aproximadamente.

En definitiva, este dispositivo recibe paquetes de tamaño mucho más grande en comparación con el tamaño de los paquetes que envía.

6. ECDF tiempos entre paquetes



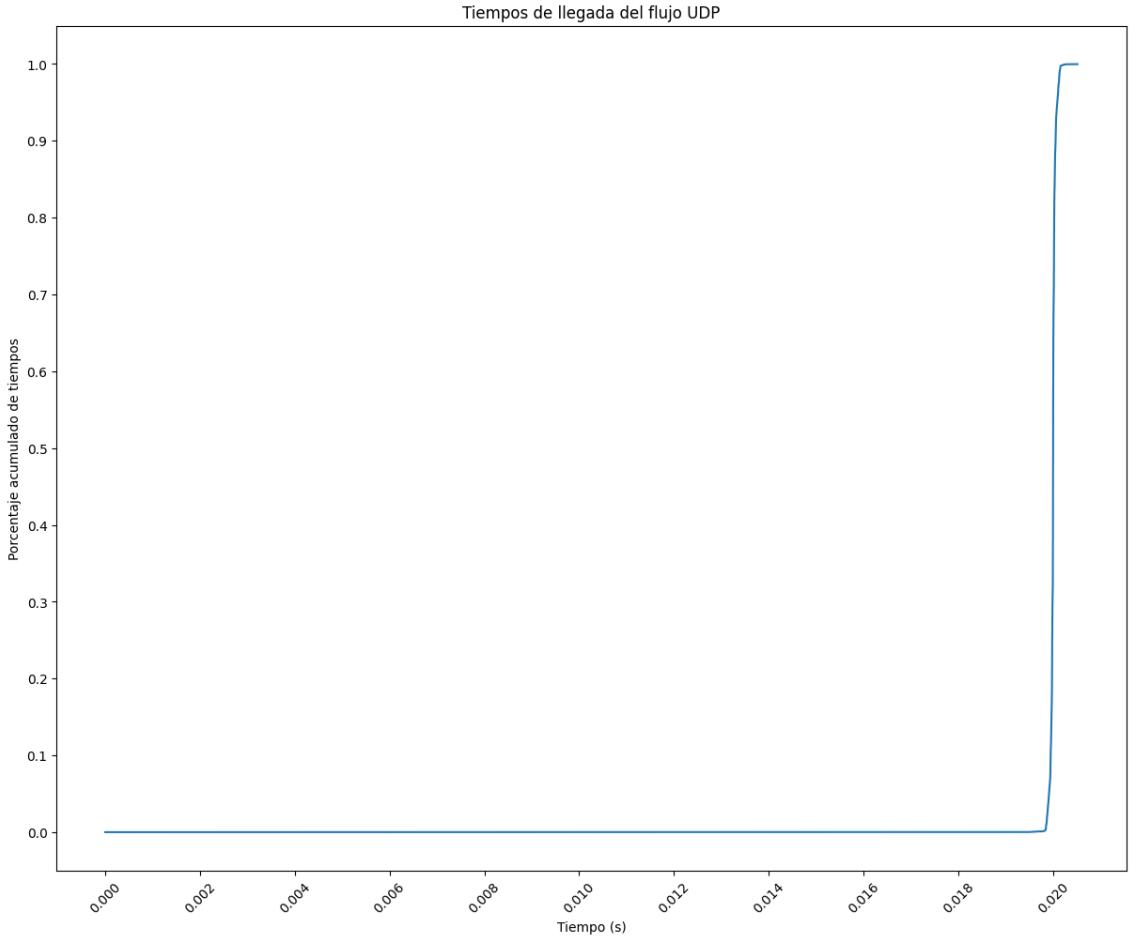
Para obtener los datos de la gráfica del ECDF del tiempo entre paquetes del flujo TCP, hemos utilizado el siguiente filtro:

tshark -r <traza.pcap> -T fields -e frame. time_delta_displayed -Y 'ip.addr eq 14.228.200.110 && tcp.srcport && tcp.dstport'

Primero filtramos aquellos paquetes que tienen dirección IP, tanto origen como destino 14.228.200.110, y que a su vez tienen puerto TCP origen y destino, y a continuación obtenemos los diferentes tiempos transcurridos desde el anterior paquete mostrado.

El protocolo TCP permite que un dispositivo pueda tener conexión a una red. Además, se encarga de que los datos lleguen sin errores, y, en orden al destino. Incluso, si durante la transmisión de los datos se perdiesen, automáticamente el protocolo TCP inicia la retransmisión. Este protocolo se usa por ejemplo para aplicaciones de mensajería.

Por tanto, en la gráfica podemos observar como este protocolo es bastante utilizado por nuestro dispositivo, ya que los tiempos entre llegadas de cada paquete son casi de 0 segundos. Por ello, podemos decir que el usuario podría estar viendo un servicio de streaming en diferido, por ejemplo, en el que es importante que la información llegue por completo.



Para obtener los datos de la gráfica del ECDF del tiempo entre paquetes del flujo UDP, hemos utilizado el siguiente filtro:

tshark -r <traza.pcap> -T fields -e frame.time_delta_displayed -Y 'udp.srcport eq 17052 || udp.dstport eq 17052'

Primero filtramos aquellos paquetes que su puerto UDP origen o destino es 17052, y a continuación obtenemos los diferentes tiempos transcurridos desde el anterior paquete mostrado.

El puerto 17052 corresponde a un puerto privado, el cual, normalmente se asigna a las aplicaciones de clientes al iniciarse la conexión.

El protocolo UDP no es un protocolo de transporte orientado a conexión, como lo es TCP. Esto quiere decir que UDP no verifica la recepción de los datos transmitidos entre dispositivos. Por ello, es más rápido que TCP, al considerar más importante que la información llegue cuanto antes, a que llegue por completo. De este modo, este protocolo es más usado para servicios de streaming en directo.

Por todo ello, podemos concluir que el usuario del dispositivo no hace mucho uso de este protocolo, es decir, no realiza tantas acciones que impliquen una llegada rápida de los datos, ya que los tiempos entre paquetes es de 0,02 segundos, bastante más grande si lo comparamos con TCP.

3 Conclusiones

Como hemos podido observar al principio de la práctica, la mayor parte de los paquetes de nuestra traza usan IP, el 91.88 %, mientras que el 8.12 % restante usa noIP. Esto tiene sentido, puesto que los dispositivos que usen otro protocolo distinto a IP (un protocolo propio) serán muchos menos de los que usen protocolo internet (IP) que es el estándar. Seguramente las máquinas que estén utilizando noIP para comunicarse entre ellas estén dentro de la misma red local.

También se puede observar cómo los datos obtenidos son coherentes entre sí, por ejemplo, el ancho de banda referido a la MAC origen se mantiene mucho más bajo con respecto al de la MAC destino y, como es de esperar, en las ECDFs de los Bytes/paquete, el tamaño de los paquetes más frecuentes es mucho más pequeño en el de origen que en el de destino como se podía intuir tras ver el ancho de banda de entrada y salida consumido por la máquina.

Finalmente, si nos fijamos en los tops de puertos UDP origen y destino, podemos observar que, para el destino, el más utilizado es el 17052 y para origen, el 24704. Podríamos suponer que el usuario asociado a esa MAC está realizando una videollamada, por ende, el primero de los puertos mencionados será el encargado de recibir la información y el segundo de enviarla.

Tras haber realizado esta práctica nos sentimos capaces de administrar un sistema de red teniendo bastante confianza en poder saber lo que ocurre dentro de él.