UNIVERSIDAD TORCUATO DI TELLA

TD IV: Redes de Computadoras

Trabajo Práctico

Segundo cuatrimestre de 2023

Autores:

CHAB LÓPEZ Catalina

CASTORE Juan Ignacio

CHEN Belén Débora

29 de Octubre de 2023

1. Introducción

En el presente informe, desarrollaremos dos herramientas clave para analizar redes. La primera es una versión personalizada de **Traceroute**, utilizando Python y Scapy. Esta herramienta sirve para trazar la ruta que hace un paquete entrante que viene desde un host o punto de red hasta tu ordenador, particularmente sirve para realizar un diagnóstico de red, ya que al enviar estos paquetes se obtienen estadísticas del RTT o la latencia de red. La segunda es un **Port Scanner** para clasificar los primeros mil puertos de un host según su estado: abierto, cerrado, o filtrado.
Al crear nuestras propias implementaciones, no solo obtendremos un mayor conocimiento sobre su funcionamiento, sino también podremos experimentar con el análisis de redes y de paquetes de datos.
A lo largo de este informe, brindaremos una explicación detallada de cómo hemos desarrollado estas dos herramientas, destacando sus características clave y sus aplicaciones. Además, compartiremos nuestras observaciones y conclusiones derivadas de los diversos experimentos que hemos llevado a cabo. El objetivo es proporcionar una visión clara y concisa de cómo estas herramientas pueden ser de gran ayuda para el análisis de redes.

2. Implementación

2.1. TraceRoute

Modo de uso: python3 traceRoute.py google.com
En la función 'traceRoute', aprovechamos la biblioteca Scapy de Python3 para construir y enviar paquetes IP. Estos paquetes se envían en un bucle, y en cada iteración, el valor del campo TTL (Time to Live) se incrementa en 1. El propósito principal de esta técnica es determinar la cantidad de saltos intermedios (routers) que un paquete debe atravesar para llegar al host de destino, que está especificado por la dirección IP proporcionada como parámetro.
El traceRoute también nos proporciona información valiosa, como las direcciones IP de todos los enrutadores intermedios que respondieron en la ruta que sigue nuestro paquete, así como un promedio de RTT entre los distintos hops.
En cada iteración, se espera una respuesta durante un tiempo limitado de 0.5 segundos. Elegimos este timeout, ya que luego de varias experimentaciones observamos que la mayoría de los routers respondían en menos de 0.2 segundos, por lo tanto 0.5 nos pareció la opción más segura y eficiente en términos de tiempo. De esta manera, minimizamos los tiempos de espera cuando tenemos que ejecutarlo muchas veces en una dirección dada, sin arriesgarnos a tener timeouts que podrían haberse evitado.
Para recopilar los resultados de las ejecuciones y facilitar el análisis de los mismos, se crea un archivo CSV en la carpeta traceroute_obs con el nombre: 'traceRoutedireccionURL.csv' y su formato de columnas es (ttl,ip,RTT). En caso de no recibir una respuesta antes del terminarse el timeout, se considera que el paquete se ha perdido en el camino. Esta pérdida se manifiesta imprimiendo '***' en la salida. En cambio, cuando se recibe una respuesta, se obtiene la dirección IP del router que respondió y se calcula el RTT restando el tiempo de envío (timeSnd) del tiempo de recepción (timeRcv). Asimismo, se verifica si la respuesta proviene de la dirección IP de destino. Si es así, se imprime un mensaje indicando que se ha alcanzado el destino, junto con el número de saltos (TTL) y el tiempo que tomó en segundos.
Para la terminación del ciclo se creó una variable 'flag' que se establece en False cuando el paquete llega a destino.

Una vez cargados los resultados en el CSV, llamamos a una función auxiliar (calculateMean) que abre este archivo y calcula 2 cosas:

I Promedio de ttl-zero-during-transit.

II Promedio de RTT a todos los hops de la ruta.

Luego agrega estos datos en forma de log (usando como ID la URL pasada por parámetro) a otro archivo llamado porcentajesRouters.csv, el cual debe estar previamente creado. Utilizaremos este archivo más tarde para sacar conclusiones.
En resumen, esta implementación de Traceroute proporciona una visión detallada de la ruta que siguen

los paquetes a medida que atraviesan la red para llegar a su destino.

NOTA AUXILIAR: Cabe destacar que al final del código, encontramos dos versiones del programa principal. La primera versión permite ejecutar Traceroute una sola vez, mientras que la segunda versión ofrece la posibilidad de especificar la cantidad de ejecuciones que se desean realizar, lo que posibilita recopilar más datos en una sola ejecución.

2.2. PortScanner

Modo de uso: python3 portScanner.py google.com (-h ó -f)
La función principal, 'portScanner', se encarga de escanear los puertos de una dirección IP de destino proporcionada como argumento. Utiliza la librería Scapy para generar paquetes TCP con el flag [SYN] activado y analiza las respuestas para determinar el estado de cada puerto.
Para determinar el estado de cada puerto, decidimos basarnos en el análisis de las respuestas a los paquetes enviados:

- Si la versión seleccionada es '-h' y la respuesta es un paquete con la flag de [SYN-ACK] activada, el puerto se considera .ªbierto". Si, en cambio, es '-f', llamamos a la función auxiliar 'sendPayload' para enviar un mensaje y verificar que se establezca la conexión TCP completa (las tres partes del 3-way handshake).
- Si la respuesta es un paquete con la flag [RST-ACK] activada, el puerto se etiqueta como 'cerrado'.
- Si no se recibe respuesta alguna, el puerto se considera 'filtrado', lo que significa que posiblemente no haya ningún proceso que esté escuchando en ese puerto, o bien que haya un firewall u otro dispositivo de red bloqueando la comunicación.
- Si se recibe una respuesta que no corresponde a ninguno de los casos anteriores, se etiqueta como 'raro' y se muestra un mensaje en la salida estándar indicando que se recibió una respuesta inesperada. Decidimos agregar este caso extra como manejo de errores.

Los resultados de este escaneo se registran en un archivo CSV y, al final del proceso, se muestran estadísticas en la salida estándar, indicando el porcentaje de puertos abiertos y filtrados.

Por otro lado, la función auxiliar, 'sendPayload', se encarga de enviar un paquete TCP a un puerto específico con un payload "holaMundoz verificar si el puerto está abierto. Esta función se utiliza dentro de la función principal para simplificar la lógica del código al escanear puertos que requieren un segundo mensaje con payload adicional.

ANEXO:
A partir de la observación de que el portScanner toma demasiado tiempo porque envía 1000 paquetes de manera secuencial, y la mayoría de los puertos no responde, superando el timeout. Por lo tanto, decidimos implementar un version auxiliar del portScanner, la cual usa threads para enviar y esperar todos los paquetes a la vez.
También declaramos una lista global respuestas = []", para que todos los threads puedan agregar una dupla (puerto, resultado), y luego al finalizar el programa poder cargar todos estos resultados en el archivo portScanner_direccionURL.csv Este archivo, quedara con los puertos ordenados en la versión original, pero con threads se desordena y por eso es importante destacar el número de puerto.
La implementación sería la siguiente:

```
.
    ## creo la lista de threads
    threads = []
    ##creo un thread por puerto a analizar
    for port in range(1, 1001):
        # cada thread ejecuta la funcion scanPort
```

```
            t = threading.Thread(target=scanPort, args=(ipDst, port,
                version))
            threads.append(t)
        # inicio los threads
        for t in threads:
            t.start()
        # espero a que terminen
        for t in threads:
            t.join()
.
```

3. Experimentación

3.1. TraceRoute

3.1.1. Primer experimento: Porcentajes promedio de ttl-zero-during-transit

En el primer experimento, determinamos el porcentaje promedio de "ttl-zero-during-transit", que representa la proporción de routers intermedios que respondieron cuando el valor de TTL llegó a cero. Para llevar a cabo esta tarea, realizamos 30 cálculos del promedio para cada una de las universidades. A partir de estos, calculamos el promedio de los porcentajes promedio, obteniendo así un resultado que, basándonos en la interpretación frecuentista de la probabilidad, podemos decir que es una buena aproximación de la proporción de routers que contestarán cada vez que realizamos un traceroute a dicha url.

Una observación que nos interesaría destacar es que los porcentajes de respuestas ttl-zero-during-transit asociados a una misma universidad tienden a repetirse. En la mayoría de los casos, encontramos que dos o tres porcentajes específicos se repetían de manera intercalada en las 30 muestras obtenidas. Esta tendencia nos lleva a suponer que probablemente se deba al uso de un conjunto limitado de 2 o 3 rutas para alcanzar el destino. Nuestra hipótesis se centra en la idea de que los servidores cuentan con un *load balancer* en su puerta de enlace (gateway) que distribuye las peticiones de manera uniforme entre las distintas instancias de servidores disponibles, evitando así la sobrecarga de una única instancia con todas las solicitudes.
En general, encontramos que frecuentemente varios routers no responden. Esto podría deberse a una variedad de factores, siendo uno de ellos que algunos routers están configurados para no responder a ciertas solicitudes por motivos de seguridad.
Particularmente en la experimentación, para la mitad de los servidores de estas 12 universidades, más del 50 % de routers intermedios no respondieron, lo que se puede ver en la tabla tabulada como "Figura 1"debajo.

Por otro lado, al llevar a cabo numerosas pruebas usando traceRoute en varias direcciones URL y utilizando distintas computadoras con conexiones WiFi diferentes, podemos inferir que, en general, los primeros cinco "hops"hacen el mismo recorrido -es decir, muestran la misma secuencia de direcciones IP- cuando se realiza desde la misma computadora con acceso al mismo WiFi. Esto se explica por el hecho de que estos primeros saltos representan la trayectoria a través de los routers proporcionados por el proveedor de servicios de Internet (ISP). En consecuencia, la dirección URL específica que se esté visitando no influye en estos primeros pasos, ya que en todos los casos se debe atravesar la infraestructura de enrutadores del ISP para acceder a Internet.

Figura 1. Porcentajes promedios de hosts intermedios que envían mensajes de ttl-zero-during-transit.

3.1.2. Segundo experimento: Análisis de latencias a universidades de distintos continentes.

En esta siguiente sección de experimentación, decidimos realizar traceroutes a dos universidades de cada uno de los continentes seleccionados.

En cada traceroute de universidades, observamos que los tiempos de respuesta (RTT) no siguen una

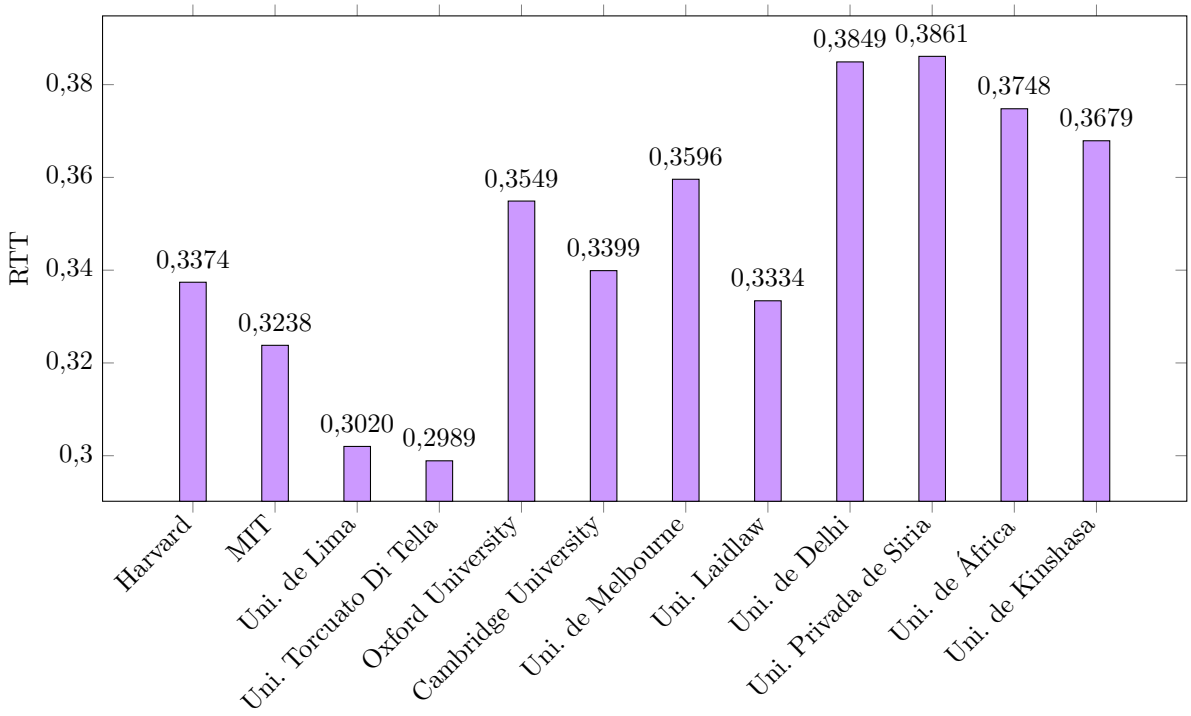
Universidad	Continente	Porcentaje promedio de ttl-zero-during-transit
Universidad de Harvard	América del Norte	47.14814815
Instituto Tecnológico de Massachusetts	América del Norte	52.88888889
Universidad de Lima	América del Sur	64.55417644
Universidad Torcuato Di Tella	América del Sur	53.06462056
Universidad de Oxford	Europa	47.26726727
Universidad de Cambridge	Europa	75.10525545
Universidad de Melbourne	Oceanía	47.87502686
Universidad Laidlaw	Oceanía	54.25577201
Universidad de Delhi	Asia	44.61604137
Universidad Privada de Siria	Asia	49.03714331
Universidad de África	África	49.75702976
Universidad de Kinshasa	África	60.97222222

progresión lineal hop a hop. En ocasiones, el RTT aumenta entre saltos, mientras que en otras disminuye, y las variaciones en la magnitud de estos cambios también son inconsistentes. Esto sugiere un comportamiento no lineal que varía de hop a hop, posiblemente debido a cambios en las rutas. La variabilidad en las rutas resulta de los múltiples cambios que experimenta la red en cuestión de segundos, tal como el tráfico de la red, retrasos en el procesamiento de routers y fluctuaciones en las tasas de transmisión de los enlaces. Logramos llegar a estas conclusiones realizando repetidas ejecuciones del Traceroute para cada dirección, y comparando los resultados expuestos en los archivos CSV generados, en los que observamos que, en distintas ejecuciones, la ruta a una misma dirección no es idéntica, y por consecuencia tampoco los RTTs.
Entonces, los factores que inciden en los tiempos de respuesta (RTT) de salto a salto no siguen un patrón intuitivo ni están directamente relacionados con la distancia geográfica.

Con el propósito de corroborar esta observación, optamos por recopilar 30 promedios de RTT a partir de diferentes ejecuciones para cada universidad, y luego un promedio general en cada caso.
Los resultados revelaron ciertas irregularidades. Notamos, por ejemplo, que Harvard y MIT, ambas ubicadas en Massachusetts (EE.UU.), presentaron diferencias en los RTTs promedio. Además, algunas universidades en el continente africano, a pesar de su proximidad geográfica –en comparación con las demás–, tenían RTTs promedio más altos que otras. Aún así, la Universidad de Lima (Perú), la más cercana a Buenos Aires, registró un RTT más bajo, lo que llevaría a pensar que sí existe una correlación tiempo-espacio intuitiva.

En última instancia, concluimos que ni la cantidad de saltos ni el RTT son indicadores fiables de la proximidad geográfica; si bien en ocasiones existe correlación entre la distancia y los RTTs, no siempre es así. Esto podría atribuirse a la distribución de la infraestructura de red a nivel global, teniendo en cuenta que los países seleccionados disponen de distintos recursos, y a la variabilidad en las rutas que el tráfico de datos puede seguir en internet. La interconexión de routers y servidores a lo largo de la red mundial puede hacer que los datos viajen distancias inesperadas antes de llegar a su destino, generando variaciones impredecibles en los tiempos de respuesta. Estos hallazgos se presentan visualmente en la Figura 2.

Figura 2. RTT promedio por Universidad.



En base a lo dicho anteriormente, otras de las conclusiones puntuales a las que llegamos fueron:

A. Al contrario de lo que se esperaría, un valor de ttl mayor no necesariamente implica un mayor RTT.
En otras palabras, no es necesario que un router situado más adelante en la secuencia de routers intermedios de la ruta del paquete tarde más en responder que uno ubicado más cerca del origen. Por lo tanto, no encontramos una correlación evidente entre el número de saltos TTL y el delay de respuesta. De hecho, para la mayoría de las universidades el host de destino (último hop) tiene menor tiempo de respuesta que routers intermedios.
Proponemos la hipótesis de que esto puede deberse a que los routers intermedios no necesariamente están geográficamente ubicados uno más lejos que el otro en relación a su orden en la ruta, sino que esto puede variar. Además, entendemos que factores como el delay de procesamiento de los routers, la congestión de la red y la tasa de transmisión de los enlaces también influyen en esta dinámica. En resumen, no se observa una relación lineal entre estos elementos.

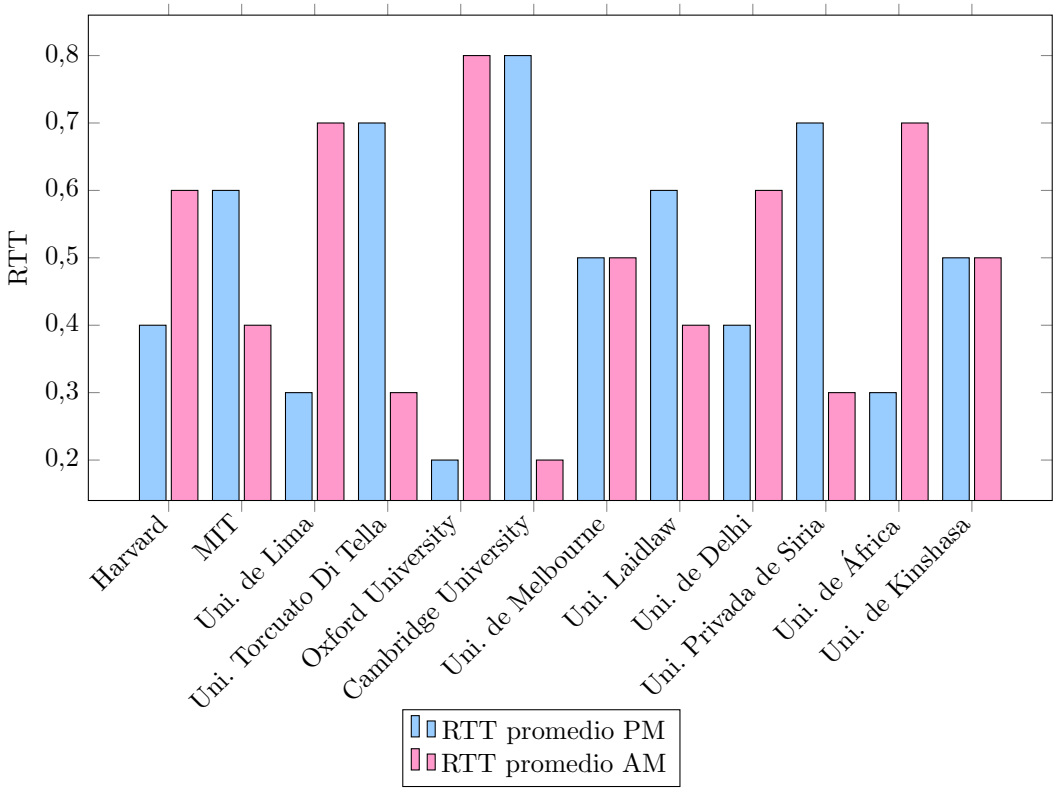
B. Mientras más hops tiene el traceroute, más varían las rutas que toma. Esto fue observado empíricamente al ejecutar el traceroute repetidamente hacia un mismo destino, generando un archivo CSV en cada ejecución para comparar las distintas rutas tomadas. Esta dinámica es razonable, ya que al tener más saltos de distancia hacia un vértice, hay más aristas posibles para seguir en el grafo. Aunque técnicamente existe un camino considerado como el "mejor.[en] un momento específico, debido a la volatilidad de la red, este camino óptimo puede cambiar rápidamente debido a las fluctuaciones en el tráfico y otros factores.

Por último, y para concluir con el análisis de los RTTs promedio, decidimos comparar los tiempos de respuesta en diferentes horarios –la madrugada y la tarde– para algunas de las universidades. Para realizar esto nos basamos en el Greenwich Mean Time[1], ya que se considera un estándar horario üniversal". Los traceroutes los realizamos alrededor de las 11 am en Argentina, que serían las 2pm según GMT y después realizamos alrededor de las 11 pm horario Argentina, que sería 2 am según GMT. Nuestra hipótesis era que durante la madrugada habría menos tráfico en la red y que debido a esto los RTTS promedio serían menores que los de la tarde.

[1]El GMT/UTC es una escala de tiempo que se utiliza como punto de referencia para sincronizar relojes y horarios en todo el mundo. A diferencia de las zonas horarias locales que pueden variar, el GMT/UTC se mantiene constante y no está sujeto a cambios estacionales, como la adopción del horario de verano.

A partir de estos datos, econtramos que no hay un patrón, pues si se observan los RTTS promedio de cada universidad se puede ver que hay únicamente 5 casos que le dan sustento a la hipótesis. El resto de los casos se dividen en dos: por un lado, se observan dos universidades con RTTS promedio iguales para la madrugada y la tarde; por otro lado, las 5 restantes tienen un RTT promedio mayor a la madrugada que a la tarde. Por lo tanto, podemos concluir en que no necesariamente afecta el horario, en particular, el horario universal. Esto puede deberse a que, aunque el horario universal sea 2 am, en cada país es un horario distinto y si haces saltos intercontinentales lo que afecta es el horario particular del país más que el universal.

Figura 3. Promedios RTT por la noche (PM) y por la mañana (AM) de diferentes universidades.



Todos los cálculos previos relativos al tiempo de respuesta se llevaron a cabo con la computadora ejecutando Zoom simultáneamente. Surgió la sospecha de que esto podría estar afectando los tiempos de respuesta debido a la posible congestión de la red a nivel local. Por lo tanto, decidimos también comparar los tiempos de respuesta cuando Zoom está en ejecución y cuando no se están ejecutando otros procesos, en el mismo momento del día.
Observamos que la diferencia entre el promedio calculado con Zoom ejecutándose (0.3042299128) y sin la aplicación en funcionamiento (0.3029835188) es de milésimas. Sin embargo, a pesar de ser mínima, también vimos que es ligeramente más rápido sin la aplicación en ejecución, como era de esperar. En consecuencia, hemos descubierto que, en realidad, la influencia de Zoom en los tiempos de respuesta es mínima.

3.1.3. Tercer experimento: Servidores de google.

En esta instancia, hemos mapeado la ruta hacia varios servidores de Google de distintos países, cambiando los dominios (.ar, .co, .uk, etc.) y notamos que el tiempo que toma alcanzarlos es casi uniforme. Esto podría atribuirse al hecho de que las direcciones IP finales están alojadas en los mismos centros de datos. Esta observación, basada en experimentación, ilustra cómo la "nube.[en] realidad se encuentra en una ubicación física centralizada. Geolocalizando las direcciones IP destino que nos devuelven las ejecuciones de traceroute para distintos dominios de google correspondientes a distintos países, observa-

mos que todos se encontraban en Estados Unidos. Por lo tanto, llegamos a la conclusión de que google probablemente tenga un datacenter destinado a todos los dominios de distintos países.
Por otro lado, al investigar la ubicación de las direcciones IP de destino de las universidades que analizamos, notamos que en su mayoría no se ubican en el país de origen de la universidad, sino que suelen estar en otros países –principalmente Estados Unidos. Una suposición razonable es que podría existir un datacenter en EE.UU. que alberga múltiples direcciones de universidades internacionales. Sin embargo, no todas las direcciones IP destino pueden ser rastreadas hasta allí. Por ejemplo, la dirección IP destino de la Universidad de Kinshasa se encuentra en Francia. Esto podría estar relacionado con motivos políticos, dado el vínculo entre Francia y la República Democrática del Congo, aunque no podemos confirmarlo.

De esta manera, a partir de esta información volvemos a confirmar que no siempre existe una correlación directa entre la distancia geográfica y el RTT promedio. Al ejecutar traceroute para una universidad en un país específico, no garantiza que obtengamos una dirección IP destino en ese mismo país.

3.2. PortScanner

3.2.1. Primer experimento: Porcentaje de puertos cerrados, abiertos y filtrados de universidades.

En este apartado, analizamos los resultados de escanear los estados de los primeros mil puertos de tres universidades distintas. Observamos que se destaca una tendencia general en la proporción de puertos abiertos, que se mantiene constante en los escaneos de la mayoría de las páginas web en un 0.2, mientras que el resto se clasifican como puertos filtrados. En particular, son los puertos 80 y 443, que corresponden a HTTP y HTTPS, los que aparecen abiertos, indicando la presencia de servicios web accesibles.

Sin embargo, una excepción notable es la Universidad de Melbourne, donde observamos un comportamiento completamente opuesto. En este caso, la mayoría de los puertos se muestran como abiertos. Esto puede ser una señal de que una amplia gama de servicios están disponibles y funcionando, pero también podría ser una consideración importante en términos de seguridad y configuración. En este sentido, una posible teoría es que no se haya configurado correctamente el firewall en el servidor o no estén adecuadamente bloqueados los puertos.

Esta variabilidad en los resultados podría reflejar las distintas políticas de seguridad y configuraciones de red de las instituciones o entidades detrás de las URLs escaneadas. Por otro lado, es importante destacar que, a pesar de las diferencias en los resultados observados entre las versiones (-f) y (-h), la mayoría de los casos muestran resultados muy similares o incluso idénticos en ambas versiones. Esta observación parece bastante razonable, ya que estamos enviando paquetes a los servidores de páginas web, los cuales usan los protocolos HTTP y HTTPS montadas sobre TCP, por lo que siempre debería completarse la conexión de 3-way handshake.

En resumen, la interpretación de los resultados del escaneo de puertos revela patrones interesantes en términos de disponibilidad de servicios y configuración de red. Podemos concluir en que existe un patrón en los estados de los puertos de los servidores web: estos tendrán siempre abiertos dos puertos de los primeros mil, el 80 y el 443, que son los que corresponden a HTTP y HTTPs.

Por último, cabe destacar que los resultados obtenidos con *nmap* fueron idénticos a los obtenidos con nuestra propia implementación del PortScanner.
A continuación, exponemos los resultados obtenidos con gráficos de torta:

Figura 4.a. PortScanning Harvard, -f

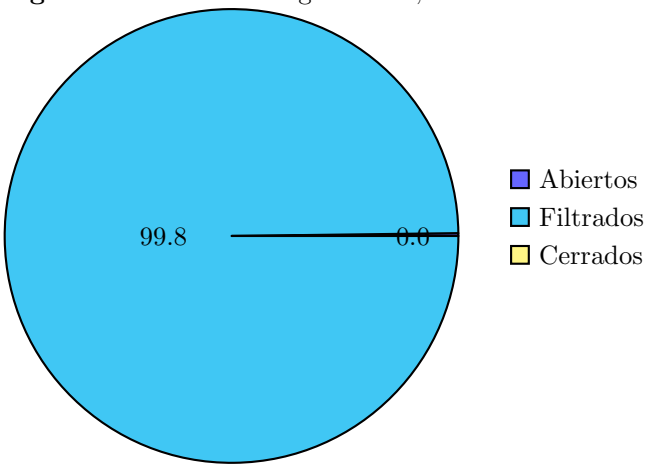


Figura 4.b. PortScanning Harvard, -h

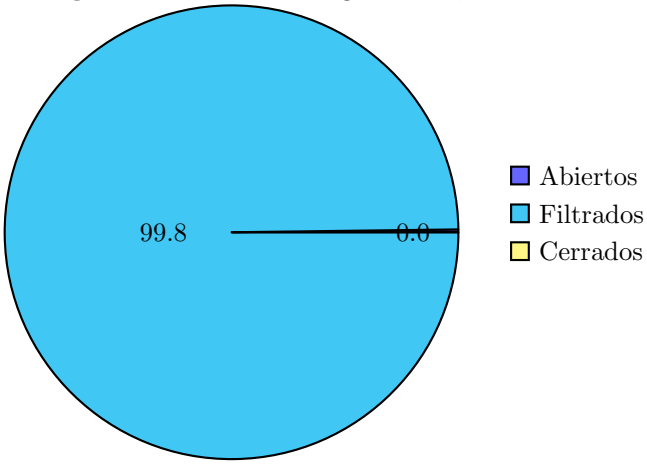


Figura 5.a. PortScanning Un. de Delhi, -f

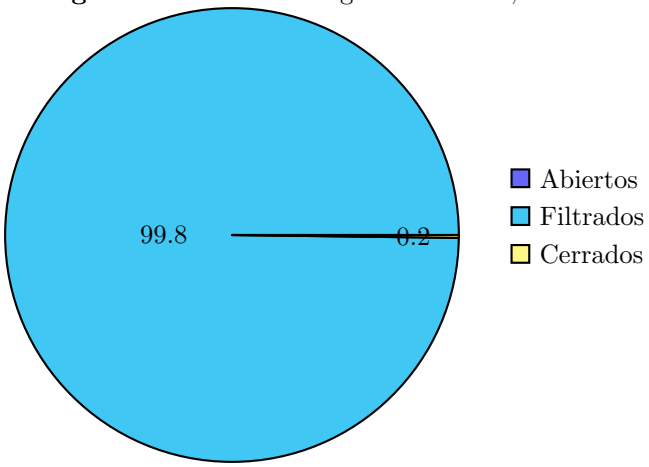


Figura 5.b. PortScanning Un. de Delhi, -h

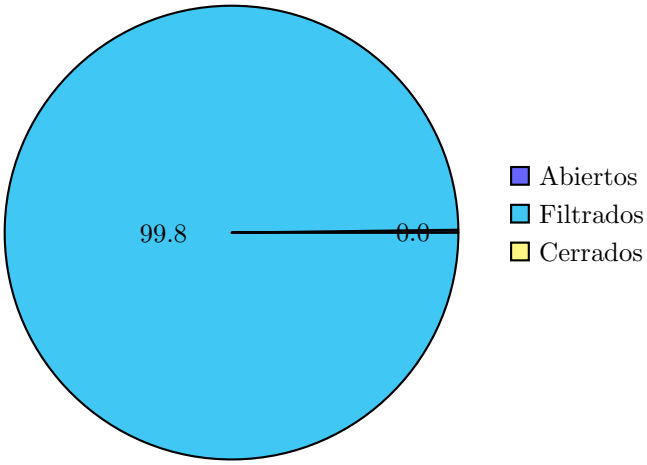


Figura 6.a. PortScanning Un. de Merlbourne, -f

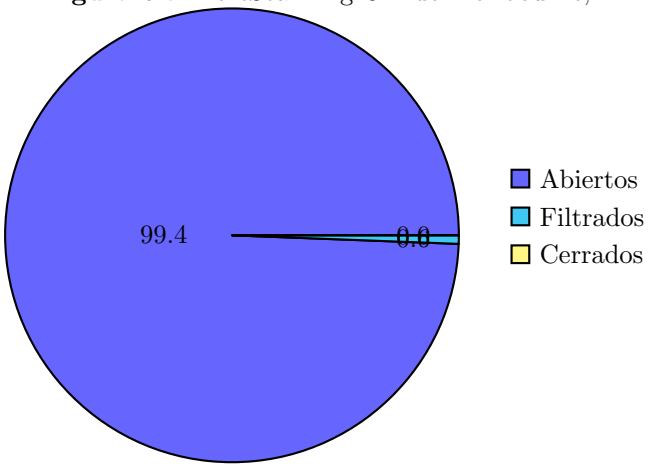
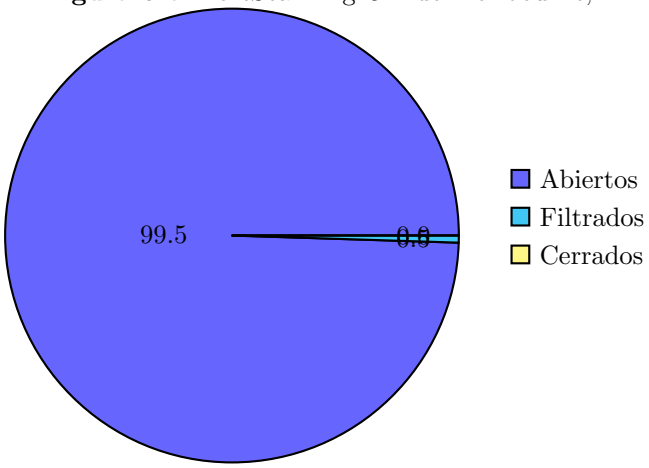


Figura 6.b. PortScanning Un. de Merlbourne, -h



3.2.2. Segundo experimento: Escaneo de servidores de correo.

Para analizar posibles patrones en el estado de los puertos de servidores de correo, utilizamos la herramienta portScanner para tres ejemplos: smtp.gmail.com, outlook.office365.com y smtp.mail.yahoo.com. Los resultados fueron diversos y dependieron de la versión seleccionada y del servidor en cuestión. Comenzando con Gmail, en ambas versiones del PortScanner se encontró el puerto 25 (SMTP) abierto. Sin embargo, en el caso de Outlook, se observaron diferencias significativas. En la versión que requiere la conexión completa TCP, no se detectó ningún puerto abierto, mientras que en la versión que solo requiere SYN ACK, se encontraron cuatro puertos abiertos: 25 (SMTP), 80 (HTTP), 110 (POP3) y 143 (IMAP). Estos mismos puertos se clasificaron como 'cerrados' en la otra versión. Por último, el servidor de Yahoo parece estar inaccesible, ya que todos sus puertos se encontraban cerrados o filtrados.

Según la herramienta *nmap*, el servidor de yahoo en cuestion estaba caído al momento del escaneo. En este experimento también comparamos los resultados obtenidos con Portscanner con los de *nmap*. Observamos que todos los resultados coincidían, excepto los obtenidos con la versión -f de portScanner para Outlook, que nos arrojaba que no había puertos abiertos cuando sí los había.

4. Conclusión

En conclusión, nuestras observaciones y experimentos resaltan la complejidad de las redes y la variedad de factores que influyen en los resultados de las herramientas que desarrollamos. No existe una lógica lineal que permita predecir con certeza cómo se comportarán las redes en todos los escenarios.

Una observación destacada en términos técnicos fue que el tiempo de ejecución de nuestras implementaciones, tanto para Traceroute como para PortScanner, fue mayor en comparación con las del sistema. Luego de investigar, podemos atribuir esta diferencia al hecho de que nuestro código está desarrollado en Python, un lenguaje interpretado, mientras que el del sistema está programado en C y C++, que son lenguajes compilados.