**UNIVERSIDAD DE LOS ANDES**

**FACULTAD DE INGENIERÍA**

**DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN**

**INFRAESTRUCTURA DE COMUNICACIONES**

**Profesor: Harold Enrique Castro Barrera**

[**hcastro@uniandes.edu.co**](mailto:hcastro@uniandes.edu.co)

**Asist. De laboratorio: Laura María Ruiz Gómez**

[**lm.ruizg@uniandes.edu.co**](mailto:lm.ruizg@uniandes.edu.co)

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **INFORME DE LABORATORIO 5 – ANÁLISIS DE COMUNICACIÓN POR SOCKETS TCP** | |  | |

**Nombres: - José Daniel Fandiño Códigos: -201423487**

* **Juan Sebastián Cardona -20142**

1. **Describa el proceso para determinar el número máximo de conexiones.**

Mediante pruebas de carga se verificó hasta cuantas conexiones podía soportar el servidor y se estableció el número máximo de conexiones en el valor máximo en donde las pruebas no tenían error. En nuestro caso eso fue alrededor de 550.

1. **Cambie el tamaño del buffer a un 10% de su tamaño inicial y realice la transferencia de un archivo, vuelva a modificar el tamaño del buffer a un 500% del tamaño inicial y** **realice la transferencia de un archivo. Describa los cambios observados, haga el análisis comparativo de las dos situaciones y concluya determinando cual es el valor adecuado para su diseño.**

Inicialmente al buffer le pusimos 8KB pues es el valor por defecto\*. Así, al probar con el 10% de dicho tamaño (es decir, 800Bytes) obtuvimos un tiempo de descarga del archivo grande de 268 segundos. Al aumentarlo al 500% (40KB) obtuvimos un tiempo de descarga del mismo archivo de 261 segundos, 7 segundos menos. También observamos que en consola para el caso del 10% los paquetes llegaban a una velocidad constante mientras que en el de 500% al inicio llegaban muy rápido pero luego se volvían más lentos. Así, es bueno tener más buffer para mayor velocidad en un inicio, pero luego puede generar latencia. En nuestro caso, como vimos que se mejoraba, decidimos subirlo a 40KB.

\*Fuente: http://www.linuxquestions.org/questions/linux-general-1/default-value-for-tcp-send-buffer-size-699906/

1. **Modifique el tamaño de los mensajes a un 10% del valor inicial y realice la transferencia de un archivo, luego a un 200% del valor inicial y realice la transferencia de un archivo, describa los cambios observados, realice el análisis comparativo de las dos situaciones y concluya determinando cual es el valor adecuado para su diseño.**

En un inicio elegimos partir los archivos en paquetes de 8KB. Al transferir un archivo con el 10% de dicho tamaño (es decir, 800Bytes), obtuvimos un tiempo de descarga de 66 segundos. Al hacerlo con el 200% (16KB), obtuvimos un tiempo de descarga de 107segundos. Al ser más pequeños los paquetes se mandan más rápido pues el TCP por debajo debe manejar pocos bytes y así hay menor probabilidad de error, y aún si llegase a haber errores reenviar el paquete no le cuesta mucho trabajo. Por el contrario, con un paquete muy grande TCP por debajo igual lo debe partir en paquetes más pequeños y asegurarse de que todos llegaron bien antes de seguir con el siguiente paquete, aumentando la latencia. Es mejor entonces quedarse con algo pequeño, pero no tanto porque igual puede ser contraproducente (pues cada paquete debe llevar su dirección y entre más paquetes entonces se transmite más datos). Para saber si valía la pena cambiarlo, comparamos también el tiempo de paquetes de 8KB y obtuvimos que se demora 89 segundos. Así, decidimos cambiar el tamaño de los paquetes a 2KB.

1. **Cambie los valores de las preferencias iniciales, describa los cambios percibidos, realice el análisis comparativo de las diferentes situaciones y concluya determinando cual es el valor adecuado para su diseño. Para esta prueba use la transferencia del archivo de tamaño grande.**

Se realizarán algunos casos para ver la diferencia. Todos se compararán con el caso de igual importancia (tiempo=latencia=ancho de banda, caso base). Casos:

* **Caso base:** Observamos que los paquetes se descargan a distintas velocidades y que la descarga se demora 210.
* **tiempo>(latencia=ancho de banda):** Aquí observamos que la descarga se realiza igual al caso base. Lo que cambia es que el tiempo de conexión disminuye, es decir, si el usuario dura inactivo se desconectará más rápidamente.
* **latencia>(tiempo=ancho de banda):** Aquí observamos que durante la descarga los paquetes se bajaban con una velocidad consistente. Es decir, los paquetes se imprimían en consola a la misma velocidad, no como el caso base donde a veces iba más rápido y otras más lento. Además, se demoró 179 segundos.
* **ancho de banda>(tiempo=latencia):** Aquí observamos que durante la descarga los paquetes van un poco más lento respecto al caso base, lo cual se ve reflejado en el tiempo de descarga de 205 segundos.

Por los anteriores resultados decidimos darle más importancia sólo a la latencia.

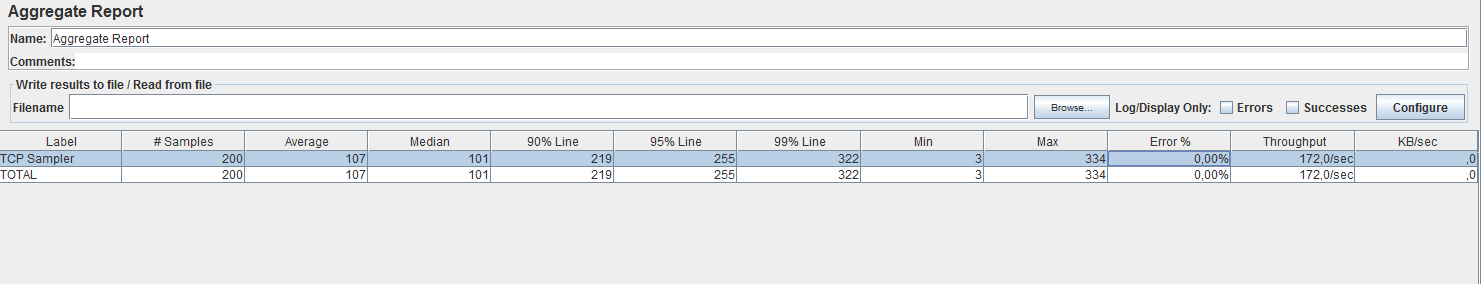
1. **Para esta parte de la práctica, se deberá hacer una descripción del diseño e implementación de cada conjunto de pruebas. Así mismo, se solicita la muestra de resultados, análisis y conclusiones para el grupo de pruebas. Apoyarse en las gráficas necesarias para sustentar lo anterior.**

Para cada conjunto de pruebas se usó JMeter, el cual nos permitía enviar una cierta cantidad de hilos para hacer pruebas de carga, además de permitirnos modificar un rampup el cual nos permitía realizar pruebas de concurrencia, y nos mandaba el tiempo que se demoraba haciendo la conexión, indicándonos el porcentaje de error de este proceso, lo que nos permitió hacer las pruebas de desempeño.

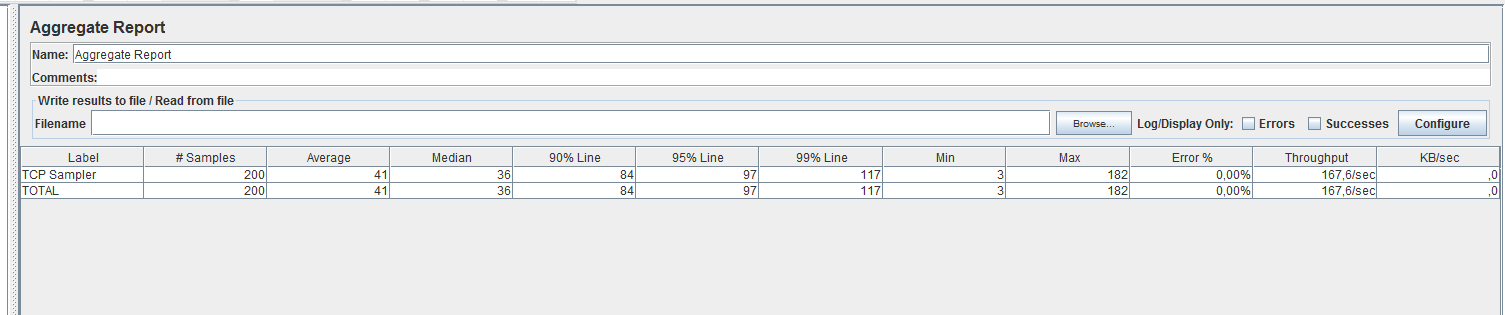
Las pruebas se hicieron en 2 tandas pruebas con un rampup de 1 segundo y pruebas con un rampup de 3 segundos.

A continuación, se anexarán los resultados.

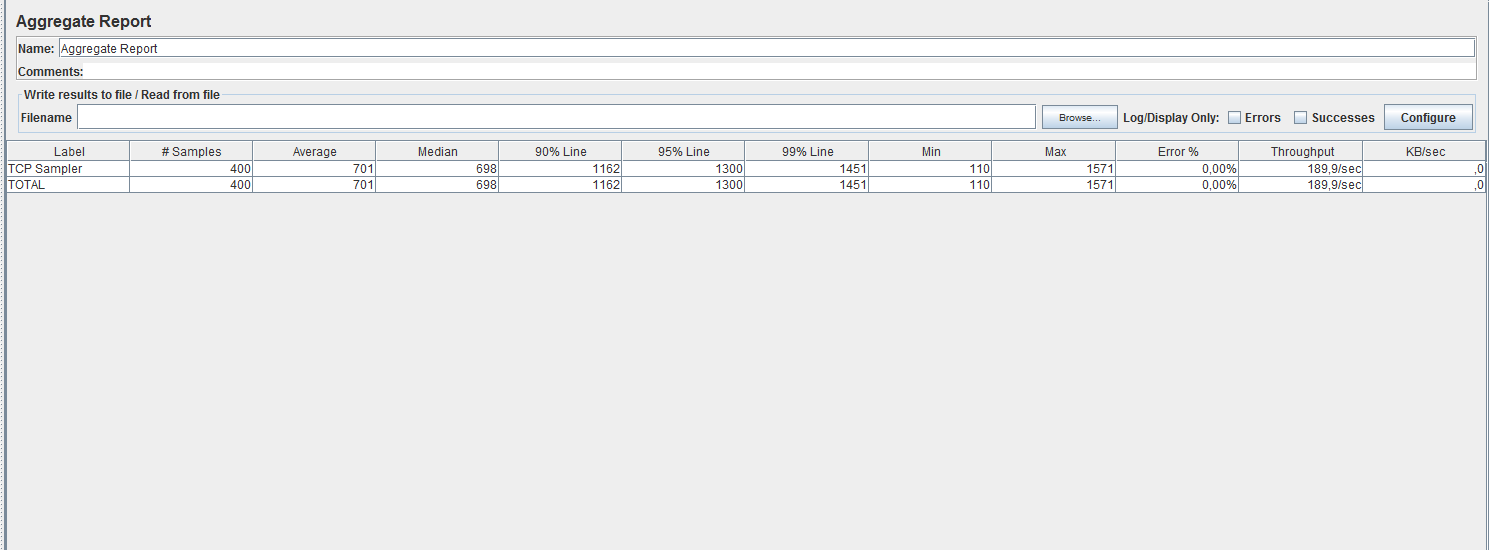
**Pruebas RampUp 1**



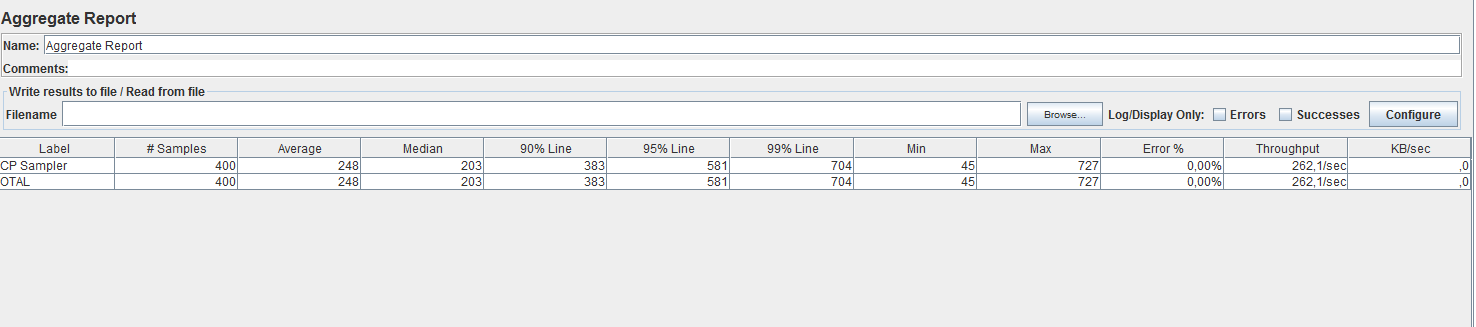
Prueba 1 200 threads

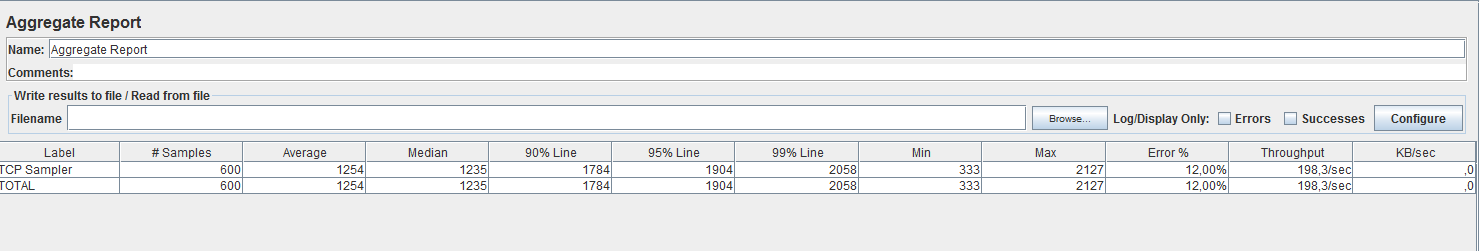


Prueba 2 200 threads

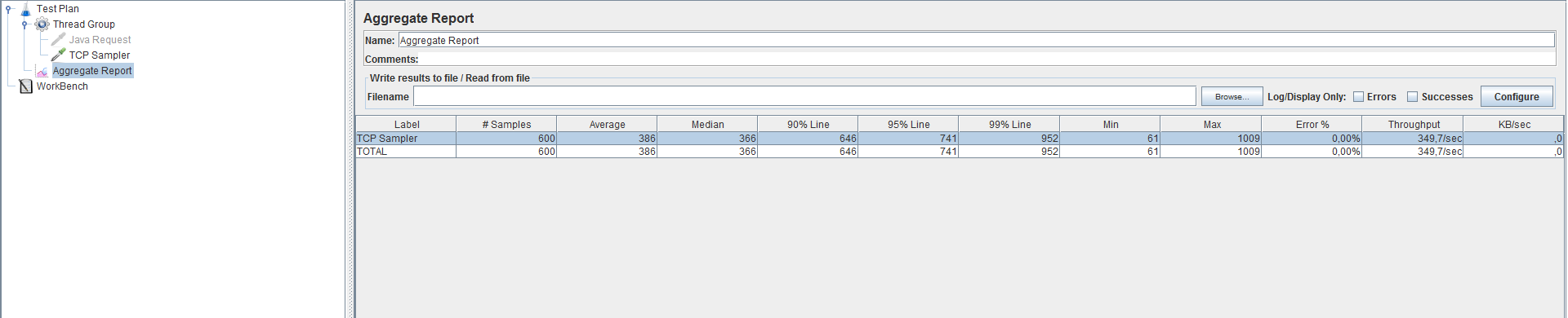


Prueba 1 400 threads

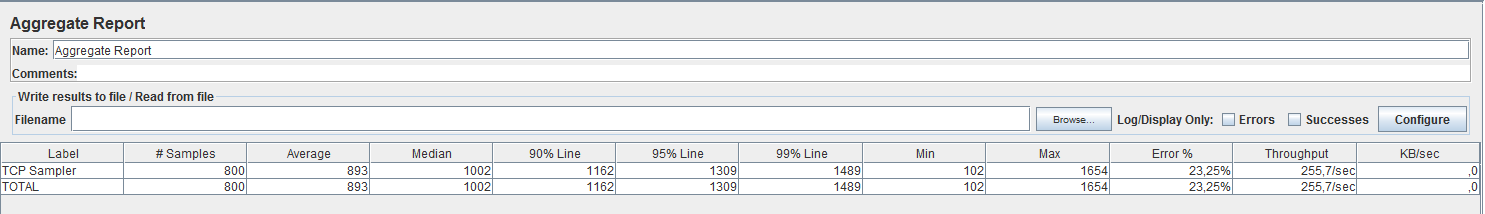
Prueba 2 400 threads



Prueba 1 600 threads



Prueba 2 600 threads

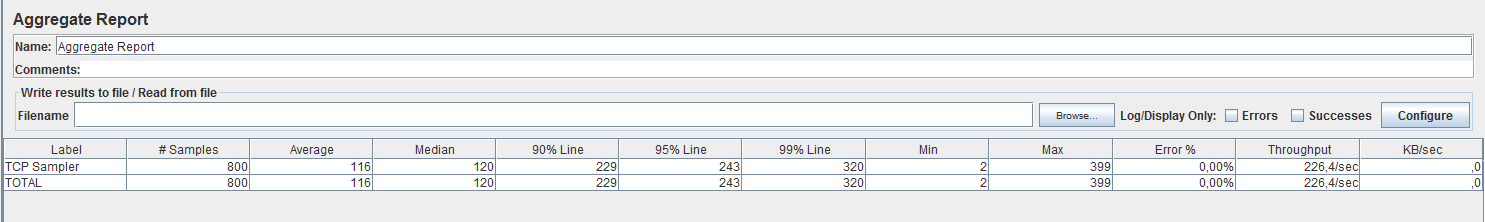


Prueba 1 800 threads.

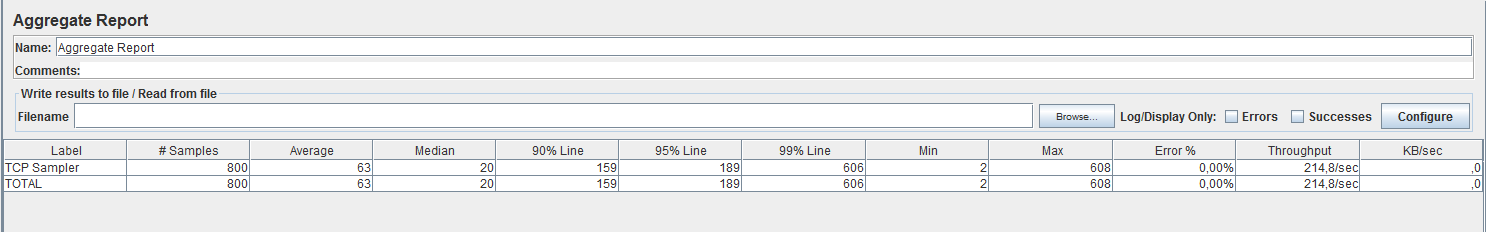
A partir de los resultados anteriores, se puede inferir que en la prueba de concurrencia alcanzaba su máximo número de usuarios entre 550 y 600. Puesto que después de 600 usuarios empezaría a sacar errores.

Por otra parte, el desempeño de la aplicación es buena antes de llegar al máximo número de estudiantes, ya que no manda error y los tiempos de respuesta son rápidos.

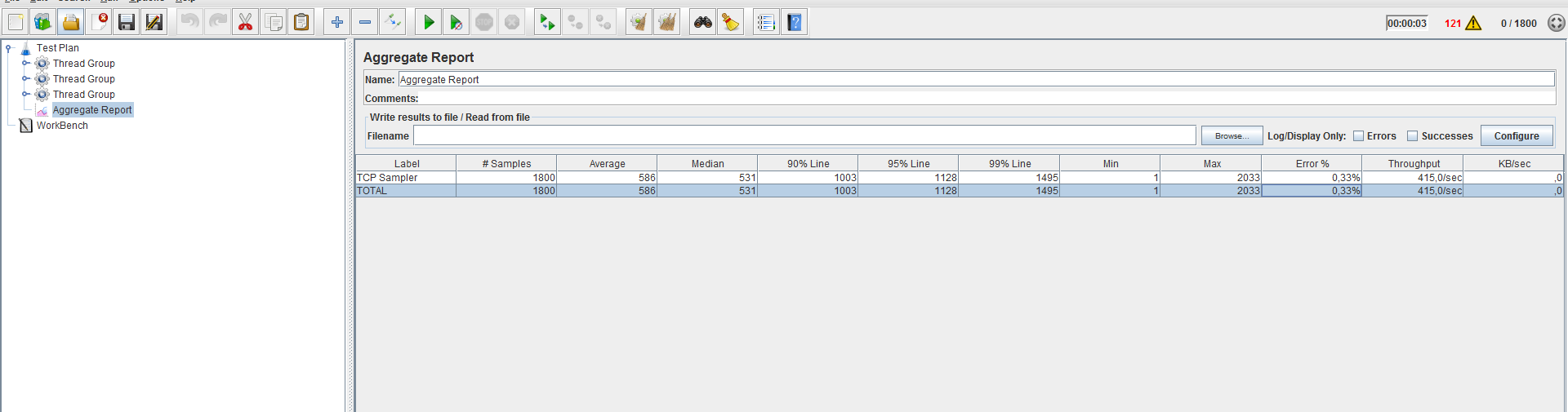
**RampUp 3 segundos.**

****

Prueba 1 800 threads

****

Prueba 2 800 threads

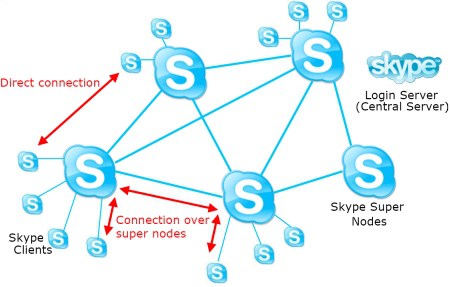
****

Prueba 1 de 1800 threads.

Con base en los resultados anteriores, se puede ver que con un rampup más amplio (3 segundos) el número de usuarios conectados al mismo tiempo (concurrencia) se multiplica, a alrededor de 1740 usuarios. Esto es debido a que la carga del servidor disminuye.

1. **Analizar las aplicaciones Waze[[1]](#footnote-1), Netflix[[2]](#footnote-2), y Skype[[3]](#footnote-3). De acuerdo a los análisis efectuados en la práctica y a referencias adicionales, argumentar cuál es el tipo de comunicación empleada por la aplicación.**

**Skype**

El protocolo de Skype corresponde a un protocolo de telefonía por internet 

punto a punto, el cual se utiliza para movilizar tráfico encriptado de carácter VoIP –voz sobre protocolo de IP- entre los computadores de los “usuarios” de Skype –puntos-.

Skype utiliza encriptación RCA para el signaling y AES para datos de voz. El protocolo –propietario- fue creado específicamente para sesiones de Skype, y no funciona con ningún otro estándar VoIP que no esté licenciado por Skype.

Skype requiere que los puertos 80 y 443 estén abiertos para el envío externo del Protocolo de Control de Transmisión, y recomienda que todos los puertos de destino sobre el 1024 se mantengan abiertos. Además, los puertos 5060 y 8000 deberían permanecer abiertos para la entrada y salida de transmisiones UDP. Skype prefiere UDP también para la transmisión de voz-.

Tomado de: https://lavideoconf.wordpress.com/2014/11/25/el-protocolo-de-skype/

**Netflix**

El protocolo de comunicación que utiliza Netflix es Discovery and Launch(DIAL).

Este protocolo tiene dos componentes, DIAL Service Discovery y DIAL REST. El DIAL Service Discovery habilita un DIAL client device para que descubran DIAL servers en su segmento de red local y obtener acceso el DIAL REST Service en esos dispositivos. El DIAL REST habilita al DIAL client a hacer querys, lanzar y opcionalmente parar aplicaciones en el DIAL Server device.

Además, se infiere que por ser un servicio de Streaming este protocolo está basado en UDP.

Tomado de:

<https://sites.google.com/a/dial-multiscreen.org/dial/dial-protocol-specification>

**Waze**

Analizando la aplicación, se podría inferir que el protocolo de comunicación de WAZE esta basado en UDP, puesto que este tiene que enviar la ruta optima lo más rápido posible, para que el usuario empiece su recorrido.

Puesto que no es estrictamente necesario un login para utilizar la aplicación, se supone que el servidor recibe request de cualquier dispositivo con la aplicación, por lo que el usuario solo haría la petición de ruta al servidor y el servidor a través de UDP le daría la ruta optima.

Las pruebas de los puntos 2, 3 y 4 se realizaron en un PC con las siguientes características:

RAM: 3,9 GB

CPU: 2,4 GHz, 2 núcleos.

Procesadores lógicos: 4

1. Página oficial: <https://www.waze.com/es/> [↑](#footnote-ref-1)
2. Página oficial: <https://www.netflix.com/co/> [↑](#footnote-ref-2)
3. Página oficial: <http://www.skype.com/en/> [↑](#footnote-ref-3)