Proyecto de Grado
Ingeniería en Computación

Descubriendo interconexiones entre Uruguay y el mundo usando tráfico popular de internet

Mateo Nogueira

Tutores:
Ing. Diego Kiedanski
Dr. Ing. Eduardo Grampín

Facultad de Ingeniería
Universidad de la República
1 de febrero de 2019

Resumen

Los servicios de transmisión de videos online son responsables de gran parte del tráfico de internet. El estudio del funcionamiento de estos servicios puede servir para brindar información sobre como se conecta nuestro país con el resto del mundo.

Este trabajo se plantea como objetivo el estudio de tráfico generado al reproducir videos de la plataforma YouTube para comprender su funcionamiento y el origen de los mismos. Con este fin, se desarrolla un programa que navega por la plataforma, recolectando y almacenando datos obtenidos al reproducir videos. Este programa es utilizado en un experimento en el cual se dejan varios equipos ejecutándolo sin parar durante el transcurso de una semana. Además, fue necesario asignar una ubicación física a cada servidor encontrado, por lo que se estudiaron y compararon diversas técnicas de geolocalización para poder seleccionar una que fuera adecuada a la realidad del proyecto.

Como resultado del experimento se logra obtener un conjunto de datos con información sobre la reproducción de casi sesenta y ocho mil videos. Se consiguió encontrar la ubicación física de los servidores de los cuales provino el contenido, se estudió la relación entre un video y sus servidores, se investigó la existencia de una jerarquía de servidores de contenido y se analizan posibles atributos que YouTube tiene en cuenta a la hora de cachear contenido.

Palabras clave

YouTube, Redes de distribución de contenido, Google, Geolocalización, Análisis de tráfico, Análisis de datos, streaming

Índice general

1. Introducción **5**
1.1. Objetivos 5
1.2. Estructura del informe 5

2. Revisión de antecedentes **6**
2.1. Redes de distribución de contenido 6
2.2. Funcionamiento de YouTube 8
2.2.1. Investigaciones Previas 8
2.2.2. Origen del contenido 9
2.2.3. Selección de servidor 10
2.2.4. Arquitectura de YouTube 11
2.2.5. Conclusiones 12
2.3. Geolocalización 12
2.3.1. Constraint Based Geolocation 13
2.3.2. GeoPing 15
2.3.3. Limitaciones 16
2.3.4. RIPE Atlas 17
2.4. Infraestructura de Google 17

3. Experimentación **20**
3.1. Geolocalización 20
3.2. Recolección de datos 23

4. Resultados **27**
4.1. Descripción del conjunto de datos 27
4.1.1. Atributos 27
4.1.2. Análisis de los atributos categóricos 29
4.1.3. Análisis de los servidores localizados 30
4.2. Agrupación de dominios 34
4.3. Grafo de Redirecciones 35
4.3.1. Jerarquía 35
4.3.2. Análisis de la jerarquía por país 36
4.3.3. Análisis de la cantidad de bytes enviados por cada país 38
4.3.4. Relación entre el idioma de un video y sus servidores de contenido . . . 40
4.3.5. Funcionamiento del stream 41

4.4. Algoritmo de recomendación y cacheo 43

5. Factores que influyen en la redirección de un video **45**
5.1. Popularidad 45
5.2. Información sobre la red 47
5.3. Metadatos sobre los videos 49

6. Conclusiones **53**
6.1. Trabajo Futuro 54

A. Implementación algoritmo de geolocalización **58**

B. Implementación del cliente **60**

C. Implementación del servidor **63**

Capítulo 1

Introducción

1.1. Objetivos

Los servicios OTT (Over-The-Top) son uno de los mayores responsables del tráfico global, representando video streaming el 73 % del tráfico de Internet en 2016 [1]. Es natural esperar que, al estudiar de dónde, cuándo y cómo proviene el contenido, se pueda arrojar luz sobre cómo nuestro país hace uso de Internet. Se decide enfocar el estudio a solamente tráfico proveniente de YouTube. Esta elección está asociada con el hecho de que esta plataforma contiene una gran cantidad de contenido y es de acceso libre. Además, existen trabajos previos que investigaron su funcionamiento y arquitectura, que pueden facilitar la comprensión de la plataforma.

El objetivo principal de este trabajo es analizar y comprender el origen y comportamiento del tráfico de YouTube. Para realizar dicho objetivo, es necesario obtener un conjunto de datos con información recabada al reproducir videos. Por lo tanto, un objetivo secundario consiste en obtener dicho conjunto de datos. Algo necesario para comprender el tráfico es conocer el origen geográfico del mismo. Una consecuencia de esto, es que se obtiene un tercer objetivo, investigar técnicas para obtener la ubicación física de un servidor y aplicar una en la práctica.

1.2. Estructura del informe

El resto del informe presenta la siguiente estructura: en el capítulo 2 se presenta la definición y funcionamiento de las redes de distribución de contenido junto con un análisis del funcionamiento de YouTube según la bibliografía encontrada, se mencionan técnicas de geolocalización y se concluye con información sobre la infraestructura de Google. El tercer capítulo expone el experimento realizado, explicando que fue implementado, duración del mismo y la técnica de geolocalización utilizada. Luego, el capítulo 4 presenta y analiza los resultados encontrados. En el capítulo 5 se analizan los datos buscando formas de identificar redirecciones del usuario al mirar un video. Finalmente, en el último capítulo se presentan las conclusiones y trabajo futuro.

Capítulo 2

Revisión de antecedentes

Este capítulo tiene el objetivo de brindar información necesaria para comprender el funcionamiento de YouTube y explicar en detalle herramientas que se van a utilizar más adelante. Por ejemplo, se da a conocer la plataforma RIPE Atlas, se analizan técnicas de geolocalización y se describe la infraestructura de Google.

2.1. Redes de distribución de contenido

Una CDN (Content Distribution Network o red de distribución de contenido) es una red de servidores con el objetivo de distribuir contenido multimedia a usuarios, intentando brindar una muy buena experiencia a ellos. Una CDN maneja servidores distribuidos geográficamente en varios lugares del mundo, almacena el contenido multimedia en sus servidores, con varias copias de un mismo elemento en distintos servidores y al recibir una petición de un usuario se asegura de que la misma sea procesada por el servidor que provea la mejor experiencia para ese usuario.

Existen dos tipos de CDNs, conocidas como CDN Privadas y CDN de terceros. En las primeras, el dueño de la CDN es también el proveedor del contenido. Un ejemplo de este tipo es la CDN de Google que se encarga de distribuir videos de YouTube y otros contenidos. Las CDNs de terceros son utilizadas por empresas que desean distribuir contenido a los usuarios, pero no están dispuestas a desplegar infraestructura alrededor del mundo, por ejemplo, debido a los altos costos que esto puede suponer. Un ejemplo de este tipo de CDN es Akamai.

Se encuentran dos grandes filosofías al respecto de cómo decidir la localización de los servidores. La primera, conocida como *Enter Deep*, consiste en instalar clústeres de servidores en ISPs (Internet Service Providers) de acceso alrededor del mundo. El objetivo de esta aproximación es estar lo más cerca posible de los usuarios finales, reduciendo la cantidad de enlaces intermedios de forma de reducir el retardo. Un problema que presenta, es debido a que los servidores están muy distribuidos, la administración y mantenimiento se vuelven complejos.

La segunda filosofía es conocida como *Bring Home*. En este caso, las CDN crean grandes clústeres en algunos lugares clave y los conectan entre sí usando redes privadas de alta ve-

locidad. En lugar de entrar a los ISP, esta aproximación consiste en colocar los clústeres de forma que estén simultáneamente cerca de PoP (Point of presence, lugares donde varios ISP se conectan a un ISP más grande) de varios ISPs tier 1 (ISPs que se caracterizan por tener cobertura internacional, intercambiar tráfico con otros ISP tier 1 y no comprar tráfico a ningún otro ISP). Comparado con la anterior, hay menos costos de mantenimiento, pero al costo de un mayor retardo.

Una vez que los clústeres están instalados, la CDN replica el contenido entre los clústeres. No obstante, es posible que la CDN no quiera replicar innecesariamente en todos sus clústeres contenido con baja popularidad o que es poco visitado en determinado país. Una solución es utilizar una estrategia en la que el contenido es replicado en un clúster si es pedido por un cliente a un clúster que no lo tiene [2]. El contenido se solicita a otro servidor de la CDN o de un repositorio central. Una vez que el espacio de almacenamiento de un clúster se encuentra lleno, se elimina contenido accedido con poca frecuencia.

Un problema al que se enfrenta una CDN una vez que recibe una petición, es como seleccionar el mejor clúster para ese cliente y elegir un servidor físico dentro del mismo. La mayoría de las CDN utilizan el servicio de DNS para interceptar y redirigir pedidos. Por ejemplo, al consultar por el dominio www.youtube.com vamos a consultar nuestro servidor DNS local (probablemente sea un servidor perteneciente a nuestro ISP) que a su vez va a consultar al servidor autoritativo de YouTube, el cual pertenece a Google y que va a poder seleccionar a cuál clúster dirigir el pedido. Para esto puede utilizar, por ejemplo, la dirección IP de origen del pedido DNS (que en este caso es la dirección IP del servidor DNS local) y elegir el servidor que este más cercano físicamente. Sin embargo, el clúster más cercano físicamente podría no ser lo más cercano en el camino en la red que recorren los paquetes. Además, un usuario podría estar usando un servidor DNS distinto al de su ISP, que podría no encontrarse cerca físicamente del usuario, lo que podría hacer que el mismo sea direccionado a un servidor de contenido cerca del servidor DNS, existiendo un servidor que se encuentre más cercano al usuario. Estas afirmaciones son válidas en caso de que el servidor DNS utilizado por el usuario sea recursivo.

Una mejor alternativa consiste en utilizar IP Anycast. Esto consiste en hacer que los routers dirijan al cliente al clúster más cercano, determinado por el protocolo BGP. Las compañías CDN utilizan la misma IP para varios clústeres y envían los prefijos mediante BGP como lo harían normalmente desde todas las localizaciones de los clústeres. Como resultado, a los routers les van a llegar varios anuncios del AS de la CDN que van a interpretarlos como si se tratara de distintos caminos al mismo lugar físico y va a elegir la mejor ruta según los criterios de BGP, que probablemente sea el más cercano en cantidad de saltos. Una consecuencia de la utilización de esta técnica es que una dirección IP no va a ser útil para identificar un servidor de una CDN dado que dependiendo de en qué lugar del mundo nos encontremos, vamos a estar accediendo a un servidor físico distinto.

Además de esto, es necesario tener en cuenta otros factores como, por ejemplo, que puede

haber clústeres saturados, en cuyo caso no es conveniente que sirva contenido a más clientes hasta que este menos cargado. Por lo que es conveniente que el usuario sea dirigido a un clúster menos saturado.

2.2. Funcionamiento de YouTube

YouTube es un sitio web que fue creado en el 2005 con el objetivo de que los usuarios de internet pudieran compartir videos. En la actualidad es la plataforma más grande con este fin, contando actualmente con más de mil millones de usuarios, que son un tercio de todos los usuarios de internet. En sus comienzos utilizaba sus propios datacenters localizados en EEUU y, algunos contratados a terceros. Sin embargo, a fines del 2006 Google compró YouTube y reestructuro la infraestructura. En el 2011 utilizaba mayoritariamente la CDN de Google y en pocas ocasiones otras CDN de terceros [3].

Para mirar un video, el usuario tiene dos formas de acceder. Es capaz de acceder ingresando directamente a la URL de un vídeo o llegar hacia él entrando por la web www.youtube.com. En cualquiera de los dos casos, un servidor de frontend provee contenido estático como imágenes y JavaScript, que son cargadas en el navegador. Además, se envía una dirección (URL) de un servidor de contenido que va a proveer el vídeo, distinto al que sirve el frontend web. El navegador consulta a su DNS local por ese servidor de contenido e inicia una conexión para pedirle el video. Sin embargo, puede ocurrir que el servidor de contenido no tenga el video disponible, por ejemplo, por ser poco popular, por lo que el servidor va a redireccionar al usuario a un nuevo servidor de contenido. El video es enviado por partes en varias peticiones.

El diagrama en la Figura 2.1 muestra la secuencia de pasos mencionada, para un caso en el que ocurre una redirección del usuario y un caso en el que no sucede. En negro se encuentran las peticiones comunes a ambos casos. En azul se ve el caso típico, donde el primer servidor contactado envía el video. Los mensajes en rojo ocurren cuando el usuario es redireccionado a un nuevo servidor. Hay que tener en cuenta que el diagrama llega hasta el momento en que se recibe la primera respuesta con contenido.

Cada video se encuentra identificado por una cadena de 11 caracteres, llamado *video_id*, por ejemplo **y9LlnLTH87U**.

2.2.1. Investigaciones Previas

Existen dos investigaciones previas que analizan el funcionamiento interno de YouTube. Datan de los años 2011 [3] y 2012 [4]. Ambas capturan datos de reproducciones de videos de YouTube e investigan su funcionamiento. En el primero, los autores capturan tráfico desde cinco lugares del mundo distintos (ubicados en Estados Unidos y Europa), en tres países distintos, y forman cinco datasets. Los puntos de captura son PoP (Points-of-Presence) de ISP nacionales y campus de universidades. Obtienen datos de 2.4 millones de videos, vistos por más de 37.000 usuarios. La segunda investigación utiliza una aproximación diferente. Los

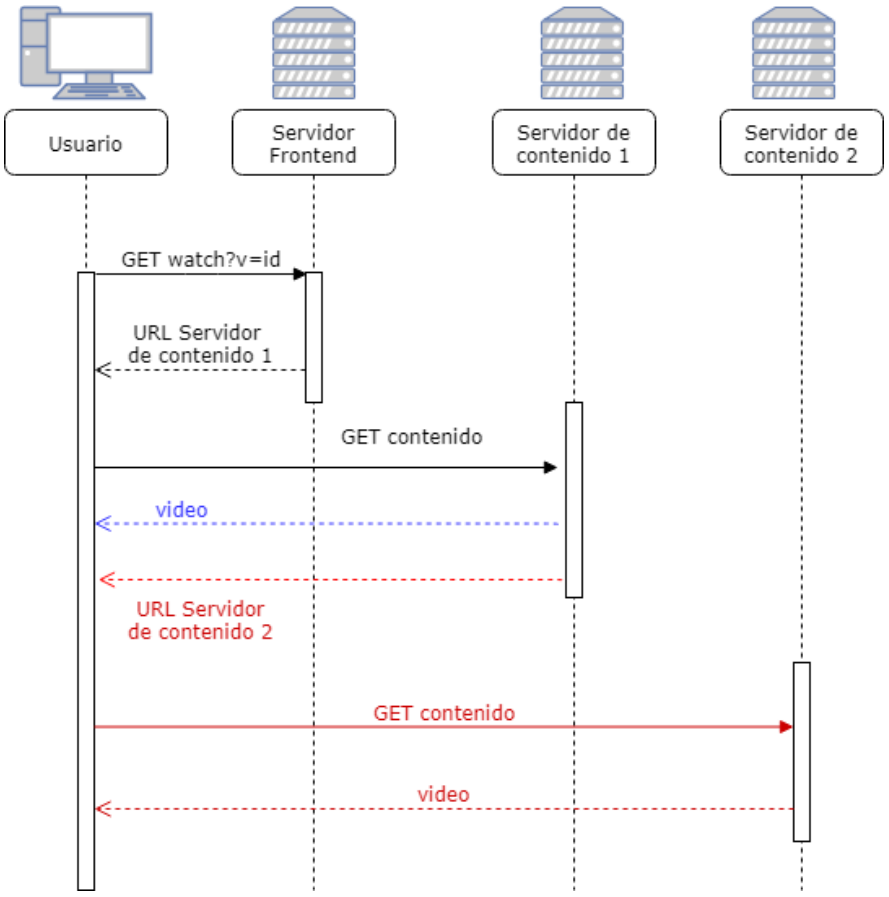

Figura 2.1: Diagrama de secuencia al solicitar un video a YouTube. Las peticiones en rojo ocurren en el caso de haber redirecciones, las indicadas en azul son el caso típico. Se omiten los mensajes de tipo DNS. El diagrama llega hasta el momento el que se recibe la primera respuesta con contenido multimedia.

autores generan un programa que simula ser un usuario navegando por YouTube que además de navegar, captura el tráfico, y realiza algunas mediciones de RTT (Round-Trip-Time).

2.2.2. Origen del contenido

El resultado del experimento realizado por los autores del artículo [3], muestra que el 82.8 % de las direcciones IP utilizadas por YouTube pertenecen al AS 15169, perteneciente a Google, y que sirvió el 98.66 % del tráfico. El que ocupa el segundo lugar es el AS 43515, cuyo dueño es YouTube EU. Esto evidencia que luego de la compra por Google, se comenzó a migrar el contenido alojado en servidores de terceros a los servidores de Google. Finalmente, detectan tráfico que proviene de direcciones IPs pertenecientes al mismo AS que el usuario mirando el video. Mostrando que Google utiliza cachés adentro de esos ISP.

Los autores geolocalizan los servidores encontrados, obteniendo así las ubicaciones geográficas y agrupan en datacenters los servidores que se encontraban en la misma ciudad. Descubren 33 datacenters: 14 en Europa, 13 en Estados Unidos y 6 en otros lugares del mundo. Se menciona que es probable que no sean todas las ubicaciones donde existan servidores con videos, debido

a las limitaciones de su experimento. Lo investigado en el otro artículo mencionado, [4], muestra que estaban en lo correcto, dado que también se realiza una geolocalización, en la que se encuentran 47 ubicaciones de servidores, que se pueden ver en la Figura 2.2.

Figura 2.2: Ubicación de los servidores de contenido según el artículo [4]. Por V. K. Adhikari, S. Jain, Y. Chen, and Z. L. Zhang, *Vivisecting YouTube: An active measurement study*, 2012, Proceedings IEEE INFOCOM.

2.2.3. Selección de servidor

Como se mencionó anteriormente, uno de los grandes desafíos inherentes a las CDN es el método para seleccionar cual servidor debe enviar el contenido y, en particular, resulta interesante investigar cómo es que Google resuelve este problema. El artículo [3] investiga cómo es que los pedidos son redireccionados a un servidor y que factores pueden influenciar la decisión.

En el artículo [3], para cada uno de los datasets capturados, se descubre que existe un datacenter que envía más del 85 % del contenido. A esos datacenters los bautiza como datacenters preferidos. Se investigan dos motivos principales por los que se pueden realizar accesos a otros datacenters distintos, los no preferidos. Estos resultan ser balanceo de DNS durante períodos de alta carga, o redirecciones a nivel de capa de aplicación.

Buscando analizar la segunda causa, se toman los videos que se descargaron exactamente una vez de un datacenter no preferido. Se descubre que el 99 % de esos videos se reprodujo una vez sola en todo el conjunto de datos. Si se toma el 1 % restante, los videos que se reprodujeron más de una vez, en todos los casos ocurre que solamente la primera vez que se reprodujo se descargó de un datacenter no preferido. En base a esto, se puede concluir que los videos que rara vez se acceden pueden no encontrarse en el datacenter preferido, causando que el usuario sea redirigido hasta un datacenter que lo tenga, y luego, YouTube lo cachea en el datacenter preferido, dando como resultado que solo el primer acceso sea al datacenter no preferido, y, mostrando que las redirecciones pueden ocurrir debido a la baja popularidad de un video.

Algo más que puede suceder, es que el video se pida a un datacenter dentro de la red privada de Google y luego sea servido al usuario desde el primer servidor al que el usuario se conectó [4]. Esto se puede observar contando el tiempo desde que se recibe el ACK de YouTube para la petición GET del usuario y cuando se recibe el primer paquete de la respuesta. Se descubre que con algunos videos poco populares este tiempo aumenta considerablemente en comparación con los casos en los que el usuario es redirigido o el servidor contiene el video. Este indicio hace sospechar que el servidor demoró ese tiempo extra porque solicitó el video a un servidor de backend dentro de la red privada de Google.

2.2.4. Arquitectura de YouTube

Conociendo que hay casos en los que un servidor no contiene el video, resulta de interés investigar como decide el próximo servidor a contactar. La investigación [4] encuentra que existe una jerarquía de servidores. La misma consta de 3 niveles, distribuidos en 38 ubicaciones primarias, 8 secundarias y 5 terciarias. Existen localizaciones que pertenecen a Google y otras que pertenecen a otros ISP como Comcast o Bell-Canada.

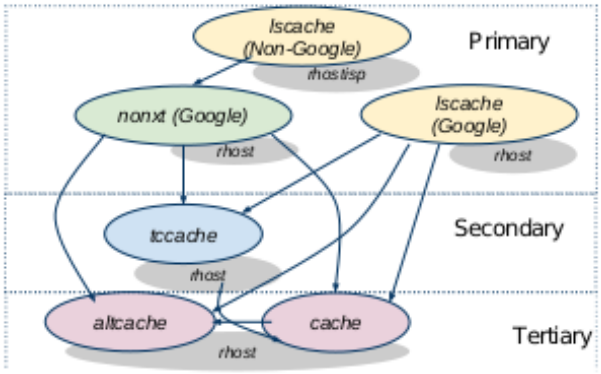

Figura 2.3: Jerarquía de servidores. En verde, azul, amarillo y rosado los namespaces anycast. La sombra gris muestra los namespaces unicast. En el primer nivel de jerarquía, se indica si los servidores pertenecen o no a Google. Por V. K. Adhikari, S. Jain, Y. Chen, and Z. L. Zhang, *Vivisecting YouTube: An active measurement study*, 2012, Proceedings IEEE INFOCOM.

Los dominios de los servidores de contenido se pueden agrupar en 5 namespaces DNS anycast y 2 namespaces DNS unicast. La Figura 2.3 muestra los namespaces encontrados junto con la jerarquía de redirecciones.

Un namespace hace referencia a todos los sub dominios pertenecientes a un dominio. Por ejemplo, los dominios *lulu.fing.edu.uy*, *proxy.fing.edu.uy*, *www.fing.edu.uy* y *ns.fing.edu.uy* pertenecen todos al namespace *fing.edu.uy*. En el contexto del artículo, un namespace anycast refiere a que un dominio perteneciente a él puede tener varias direcciones IP asociadas, mientras que, en el caso de los namespaces unicast, existe solamente una dirección IP asociada

cada dominio como máximo.

En la actualidad estos namespaces fueron reemplazados por un solo namespace unicast, *googlevideo.com*.

En la Figura 2.3 se puede observar mirando las flechas entre los namespaces, cual namespace puede redireccionar a cuál. En general, un servidor en un namespace solo puede redireccionar a otro servidor en su mismo namespace o a un servidor en un namespace de jerarquía mayor. Por ejemplo, los servidores en *lscache* redireccionan a servidores en *tccache* (un nivel mayor) o, a servidores en *cache* (dos niveles mayores).

2.2.5. Conclusiones

En el 2011 y 2012, YouTube asignaba un datacenter a cada usuario, del cual intentaba servir todo el contenido para ese usuario. Ese datacenter es el primer lugar al que el usuario le pide contenido. En caso de que el video no se encuentre en ese datacenter, el usuario o bien es redireccionado sucesivamente hasta que se encuentre un servidor que contenga el video, o, el datacenter obtiene el video desde un backend de Google y lo sirve al usuario. La popularidad de los videos parece ser un factor que influencia en la posibilidad de la redirección de un usuario.

Cabe destacar, que estos experimentos fueron realizados con equipos en Estados Unidos y Europa. No se encontraron investigaciones referentes de este tema enfocadas en Latinoamérica, y, por lo tanto, tampoco en Uruguay.

2.3. Geolocalización

Si bien las direcciones IP se reparten en bloques y se otorgan a un ISP en un país, es fácil encontrar ejemplos de direcciones IP que están asignadas a servidores que se encuentran en un país distinto al del ISP dueño de esa dirección. Un ejemplo sencillo desde nuestro país es utilizar la herramienta ping a www.youtube.com. Se puede observar que la dirección IP está asignada a Google, localizado en Estados Unidos, pero el RTT es de aproximadamente 20 ms, algo que no es posible debido a los límites físicos existentes en los enlaces. Resulta necesario entonces, idear una metodología que permita encontrar la ubicación geográfica de esos servidores, lo que se le conoce como geolocalización.

Conocer la ubicación geográfica de un equipo puede tener diversas aplicaciones. Por ejemplo, dado un usuario, mostrarle publicidad acorde al país donde se encuentra. También se puede seleccionar el idioma en el que se va a mostrarle un sitio a un usuario. Una utilidad que es relevante a este proyecto, es que conocer la ubicación de un usuario puede ayudar a una CDN a balancear la carga y enviar el contenido replicado desde el lugar más cercano al usuario [5].

A continuación, se analizan diversos métodos de geolocalización encontrados. Se considera de interés resaltar que la mayoría de estas investigaciones fueron publicadas hace muchos años. Es decir, no han ocurrido avances significativos en este tema en los últimos tiempos.

2.3.1. Constraint Based Geolocation

El primer método a analizar, se llama Contraint Based Geolocation (CBG) [6]. Este método, intenta aplicar un proceso llamado multilateración para geolocalizar servidores. La posición de un objetivo puede ser estimada usando una cantidad determinada de medidas de la distancia del objetivo a puntos cuya posición es conocida. Un ejemplo de este método es el utilizado por los sistemas GPS [7]. Los sistemas GPS utilizan tres medidas (triangulación) a satélites conocidos para estimar la ubicación de un objetivo. La clave para que esto funcione es una buena precisión en los cálculos del tiempo. Los satélites utilizan relojes atómicos que dan medidas "perfectas", dando un error muy bajo.

En internet, intentar llevar esto a cabo tiene ciertas limitaciones. Por ejemplo, las medidas no son tan precisas como en los sistemas GPS. Además, los bits viajan por cables, que no siempre siguen el camino más corto entre dos puntos del mundo. Realizando algunas suposiciones, es posible convertir una medida del retardo entre dos equipos a la distancia entre esos equipos. En general, el retardo entre dos equipos puede dividirse en dos componentes. Una componente fija y una estocástica. La primera se ve influenciada por el tiempo mínimo de procesamiento por los routers intermedios, el retardo de transmisión y el retardo de propagación. La componente estocástica se compone del retardo de cola en routers intermedios.

CBG asume que dos equipos dados se encuentran conectados por el camino más corto sobre la tierra. Luego, asumiendo además que el único retardo es el de propagación, considerando que los bits se mueven a $2/3\ C$ (siendo C la velocidad de la luz en el vacío), se obtiene una regla de 1 ms de RTT por cada 100 km de cable. Esto da una distancia máxima a la que un objetivo se puede encontrar del equipo de donde se lanza la medida y, a la recta que representa dicha relación se le llama *baseline*. Sin embargo, para considerar los otros retardos en la red, se define la *bestline*. Dado un conjunto de k equipos, que se van a utilizar para geolocalizar el objetivo, $\{L_1, l_2, ..., L_k\}$ y, un conjunto de medidas entre todos los equipos, (x, y) donde x representa la distancia entre un objetivo a y un objetivo b y, y el RTT entre a y b, se define la *bestline* como la recta que pasa lo más cerca pero, por abajo de los puntos (x, y) y, su intersección con el eje y es positiva o cero (si fuera negativa, implicaría una distancia negativa). Cada uno de los k equipos calcula la distancia a un objetivo utilizando su propia *bestline*. Si se toman las medidas entre los equipos con cierta regularidad, el algoritmo se calibra automáticamente y determina la relación entre cada uno de los k equipos y el resto en base al estado de la red y la distancia geográfica entre ellos. La Figura 2.4 muestra para un equipo, ambas rectas, junto con los valores de RTT y la distancia, desde uno de los k equipos a los otros $k-1$.

Para saber la ubicación de un objetivo, se calcula el RTT desde cada uno de los k equipos, y se estima la distancia a la que se encuentra de cada equipo utilizando la *bestline*. Este

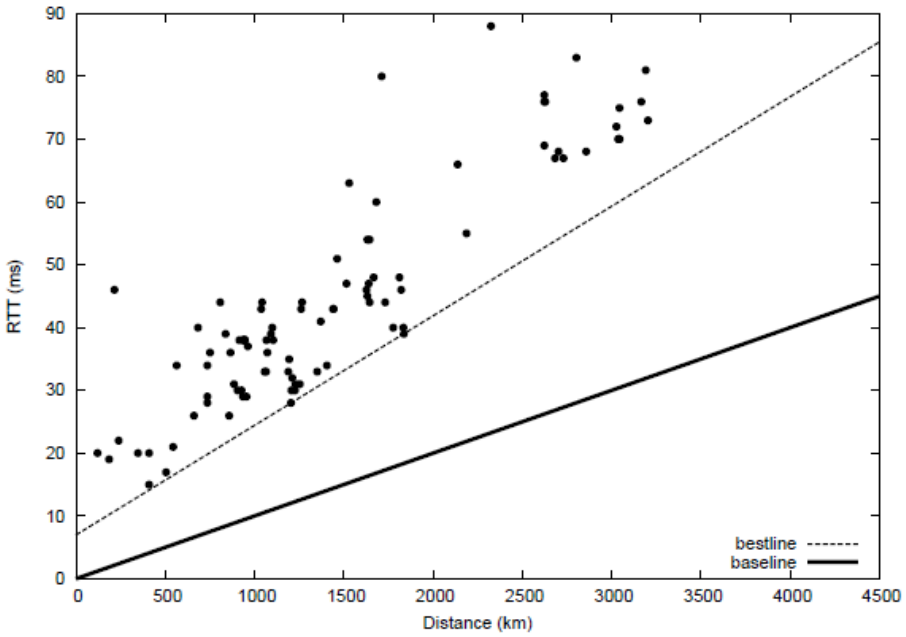

Figura 2.4: Bestline y baseline para un equipo, junto con los RTTs y distancias a los restantes equipos del conjunto de equipos a utilizar para geolocalizar el objetivo. B. Gueye y col. "Constraint-Based Geolocation of Internet Hosts". En: IEEE/ACM Transactions on Networking 14.6 (dic. de 2006), pags. 1219-1232. issn: 1063-6692. doi: 10.1109/TNET.2006.886332.

resultado es la distancia real al objetivo sumado a un error. Por ejemplo, para el equipo en la Figura 2.4, un objetivo con un RTT de aproximadamente 15 ms, según la baseline se puede encontrar a una distancia máxima de aproximadamente 1500 km, y, según la bestline, se encuentra a aproximadamente 500 km. Entonces, para cada equipo, el objetivo se encuentra en algun punto de una circunferencia que tiene como radio la distancia calculada utilizando la *bestline* y el equipo como centro. Finalmente se realiza la multilateración tomando como ubicación el centro de la región formada por la intersección de todas las circunferencias.

Se puede notar que, la idea detrás de utilizar la bestline, es crear una cota de la distancia máxima, inferior al límite físico dado por el retardo de propagación. Uno de los problemas que esto puede generar es que, si se encuentra un equipo con un RTT muy bajo, no se va a poder geolocalizar. Por ejemplo, mirando nuevamente la Figura 2.4, si el objetivo tiene un RTT menor a 8 ms, el equipo al cual pertenece la gráfica en la figura va a asignar una distancia negativa al objetivo. Otro ejemplo, es el de la Figura 2.5. Lo que sucedió en ese caso fue que las restricciones que generó el algoritmo fueron demasiado grandes, ocasionando que, la distancia resultante a la que se encuentra el objetivo para cada equipo, sea menor a la real. Sin embargo, los autores aseguran que en sus experimentos las distancias siempre fueron sobreestimadas, por lo que esta situación nunca se dio.
Los autores del artículo utilizan dos conjuntos de datos para probar su algoritmo, uno con datos en Estados Unidos y otro con datos de Europa. Obtienen un error medio de 100 km para el de Estados Unidos y 25 km para el de Europa.

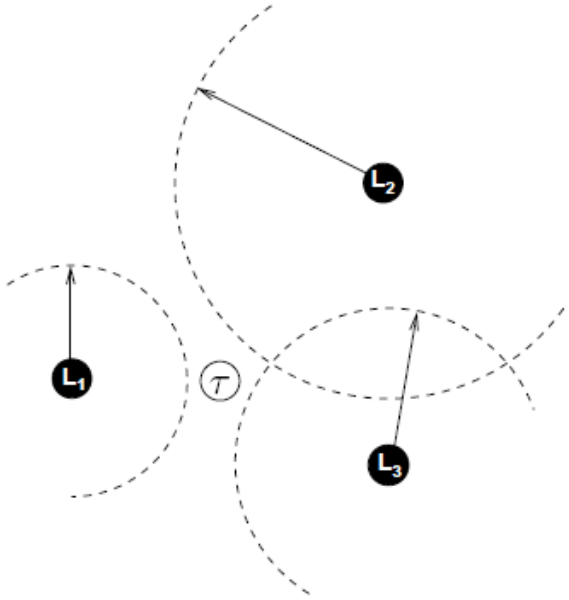

Figura 2.5: Geolocalización utilizando CBG. Las distancias se generaron cotas muy restrictivas, que no permiten geolocalizar al objetivo. B. Gueye y col. "Constraint-Based Geolocation of Internet Hosts". En: IEEE/ACM Transactions on Networking 14.6 (dic. de 2006), pags. 1219-1232. issn: 1063-6692. doi: 10.1109/TNET.2006.886332.

2.3.2. GeoPing

Se describe un segundo método llamado GeoPing [8]. Este método utiliza nuevamente un conjunto de k equipos y las medidas de retardo entre ellos. La diferencia radica en cómo se utilizan esos retardos para asignar una ubicación geográfica a un objetivo. En este caso, se construye un mapa de retardos (mapa en el sentido de la estructura de datos), donde cada entrada contiene dos elementos. Las coordenadas de un equipo conocido (uno de los k equipos) y un vector de retardos $DV = (d_1, d_2, ..., d_k)$ que contiene el retardo mínimo encontrado entre ese equipo y el resto de los $k-1$ equipos. A la hora de geolocalizar un objetivo, se calcula el retardo mínimo desde cada equipo a ese objetivo y se construye un vector de retardos $DV' = (d'_1, d'_2, ..., d'_k)$. Finalmente se calcula la distancia (euclidiana) entre ese vector DV' y todos los vectores del mapa de retardos. Al objetivo se le asigna la misma localización que el equipo cuyo vector se encontró a menor distancia.

Los autores lograron obtener un error de 150 km para el 25 % de los casos, utilizando $k = 7$. El artículo en el que se elabora CBG [6], realiza una comparación entre su algoritmo y GeoPing. Esto se puede observar en la figura 2.6. Realiza dos comparaciones. En uno de los casos utiliza equipos y objetivos localizados en Europa, y, en el otro, equipos y objetivos que se encuentran en Estados Unidos. Se aprecia que CBG obtuvo un error menor en los dos conjuntos de datos, siendo bastante mejor en el experimento realizado en Europa. Sin embargo, el error en el experimento llevado a cabo en Estados Unidos es similar para ambas técnicas. Los autores sospechan que se debe a que en Estados Unidos se utilizaron una mayor cantidad de equipos. Sin embargo, se debe tener en cuenta que no se informa cual es el valor

de k utilizado para GeoPing.

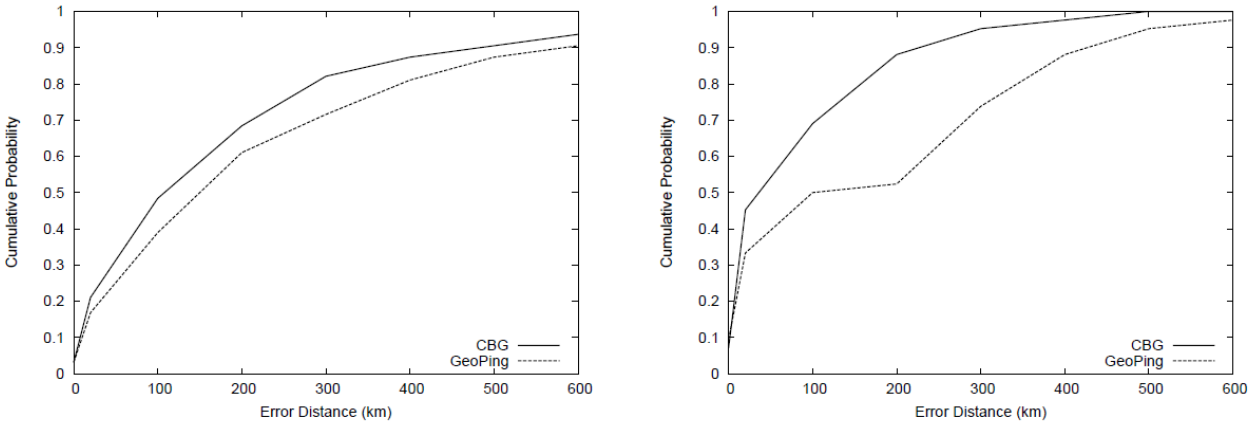

Figura 2.6: Comparación entre los errores de CBG y GeoPing. La imagen de la izquierda muestra el error de los datos recolectados en Estados Unidos, mientras que, la imagen de la derecha representa el error en el conjunto de datos recolectado en Europa. B. Gueye y col. "Constraint-Based Geolocation of Internet Hosts". En: IEEE/ACM Transactions on Networking 14.6 (dic. de 2006), pags. 1219-1232. issn: 1063-6692. doi: 10.1109/TNET.2006.886332.

2.3.3. Limitaciones

Se finaliza esta sección comentando algunas de las limitaciones en estos enfoques. Lo principal, es que se asume ciertos retardos como muy bajos, cuando en la realidad podría no ser así. Por ejemplo, los retardos de cola, estos pueden considerase despreciables siempre y cuando no exista congestión en la red y la cantidad de saltos intermedios no sea muy grande. Hay que destacar, que estos experimentos se realizaron en zonas aisladas (Estados Unidos y Europa). No obstante, en este proyecto la zona de búsqueda es mayor, porque en principio, los servidores podrían encontrarse en cualquier parte del mundo.

Otro problema es el cuello de botella que se puede generar en el enlace entre un usuario y el primer router de un ISP, en caso de que se esté conectado por una línea de baja velocidad como un modem telefónico. Este problema, que es mencionado en los artículos, es probable que hoy en día debido a los avances en la tecnología, no presente grandes dificultades.

Finalmente, pero muy importante, hay que recordar que los caminos en la red probablemente no sigan el camino más corto sobre la tierra. En el artículo sobre CBG se menciona un estudio sobre que tanto se desvían las rutas de este camino ideal. Los resultados indican que el nivel de conectividad de la red y las políticas de conexión entre los sistemas autónomos juegan un papel muy importante en esto.

2.3.4. RIPE Atlas

En los métodos de geolocalización se mencionaba el uso de equipos distribuidos para poder realizar mediciones que eran utilizadas para la geolocalización. Los equipos que se utilizarán en este proyecto van a ser probes de RIPE Atlas.

RIPE NCC [9], es una organización sin fines de lucro, encargada de administrar recursos de red (por ejemplo, direcciones IP y números de sistemas autónomos) en Europa, el Medio Oriente y partes de Asia Central. RIPE posee una plataforma global de mediciones llamada RIPE Atlas [10], que mide la conectividad y alcance, brindando información sobre el estado de la red en tiempo real.

Voluntarios alrededor del mundo conectan pequeños equipos *probes* (sondas) en sus casas u oficinas. Estas probes se encuentran funcionando las 24 horas del día, recolectando datos. Los resultados son subidos a internet y se encuentran disponibles públicamente. Las personas que instalan una probe, consiguen créditos que luego pueden usar en mediciones creadas por ellos mismos. Existen varios tipos de mediciones, entre las que resulta interesante para este proyecto la medición de tipo *ping*, que mide el RTT entre dos equipos. Las medidas se pueden programar para que se ejecuten en determinado tiempo, por cuanto tiempo, cual es el objetivo de la medida, y desde que probes se quiere realizar la medida.

Por ejemplo, si la facultad de ingeniería desea medir el RTT a su sitio web desde distintos lugares del mundo, puede utilizar una medida de RIPE Atlas, con objetivo la dirección IP o dominio del sitio web, seleccionar probes desde distintos lugares del mundo y programar la medida para que se ejecute en cierto día.

Además de las probes mencionadas anteriormente, existe otro tipo de probe, llamada *anchor*. Dichas probes funcionan de la misma manera que las otras, con la diferencia de que poseen mejores capacidades de medición y ubicadas en lugares con un ancho de banda capaz de soportar una gran cantidad de mediciones. Generalmente son desplegadas por organizaciones grandes como ISPs.

2.4. Infraestructura de Google

En su sitio web [11], Google da a conocer su infraestructura, intentando explicar cómo es que los ISP se conectan a su red. Cabe aclarar que esto no es específico a YouTube, si no que es general a todos los servicios de la empresa. Según se explica, cuando un usuario ingresa a una aplicación o sitio web sucede lo siguiente. Primero, la solicitud es enviada a un servidor de Google cercano al usuario, que va a proveer la menor latencia. Ese servidor, envía la petición al datacenter más cercano y genera la respuesta que provee *la mejor experiencia* para ese usuario en ese momento. Finalmente, la respuesta es descargada por el usuario. La red está compuesta por tres grandes componentes.

- Un conjunto de datacenters.

- Un conjunto de **Edge Point of Presence (PoPs)**.
- **Edge Nodes**, conocidos como **Google Global Cache** o **GGC**.

Los datacenters son 15 y están distribuidos alrededor de todo el mundo. La Figura 2.7 muestra un mapa con la ubicación de cada uno. Se pueden encontrar ocho en Estados Unidos, uno en Chile, cuatro en Europa y dos en Asia. Todos los datacenters se encuentran conectados por una red privada construida por Google.

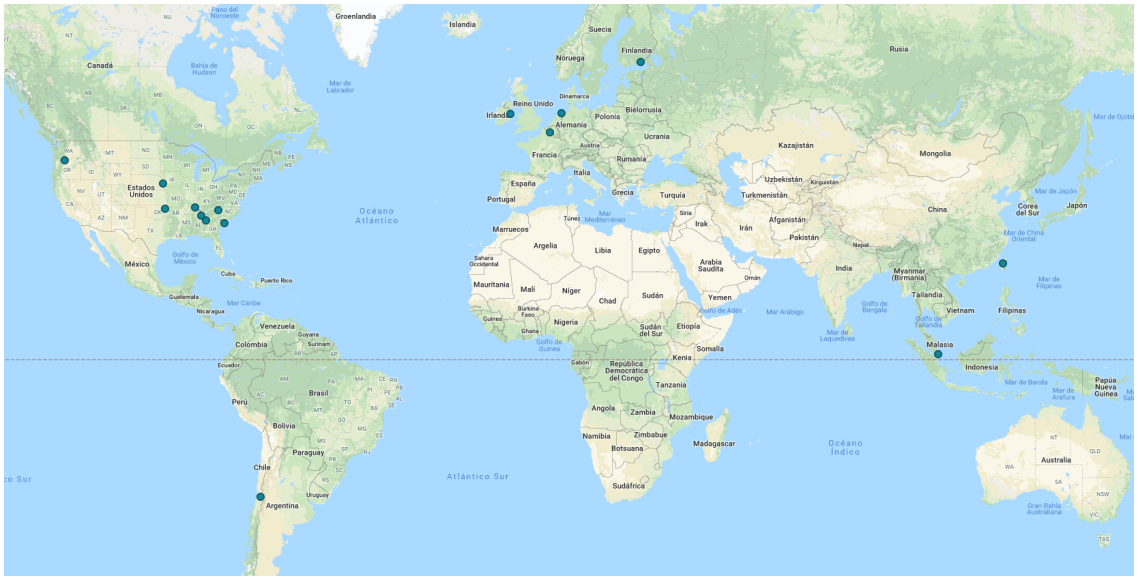

Figura 2.7: Mapa con las ubicaciones de los datacenters de Google.

Los PoPs son los puntos donde la red privada de Google se conecta con el resto del mundo. Los ISP se pueden conectar con Google e intercambiar tráfico en más de 100 lugares alrededor del mundo. Los más cercanos a Uruguay se encuentran en Santiago de Chile, Buenos Aires, San Pablo y Rio de Janeiro. Se puede observar el conjunto total de points of presence en la Figura 2.8.

Finalmente, los GGC son la parte de la infraestructura más cercana al usuario. Permiten a los ISP desplegar servidores de Google adentro de sus redes. En particular, se menciona que el contenido estático (como YouTube o Google Play) popular se cachea temporalmente en esos servidores. Se pueden encontrar por lo menos uno en casi todos los países del mundo, tal como se puede ver en la Figura 2.9. El más cercano al país, se encuentra adentro de él. El mapa que se puede encontrar en el sitio mencionado anteriormente lo marca adentro del aeropuerto de Carrasco.

En caso de querer conectarse a la red de Google, un ISP tiene varias opciones que puede tomar. Entre ellas, cabe destacar la posibilidad de instalar un GGC en su red (algo que parece que ya fue hecho por Antel) y conectarse con los sistemas autónomos AS 15169 o AS 36040, pertenecientes a Google. Las conexiones están disponibles en ciertos lugares, que se pueden encontrar en el sitio PeeringDB [12]. Entre ellos, para el AS 15169 hay una ubicación en Miami, que, según el mismo sitio, Antel también está conectado en ese lugar.

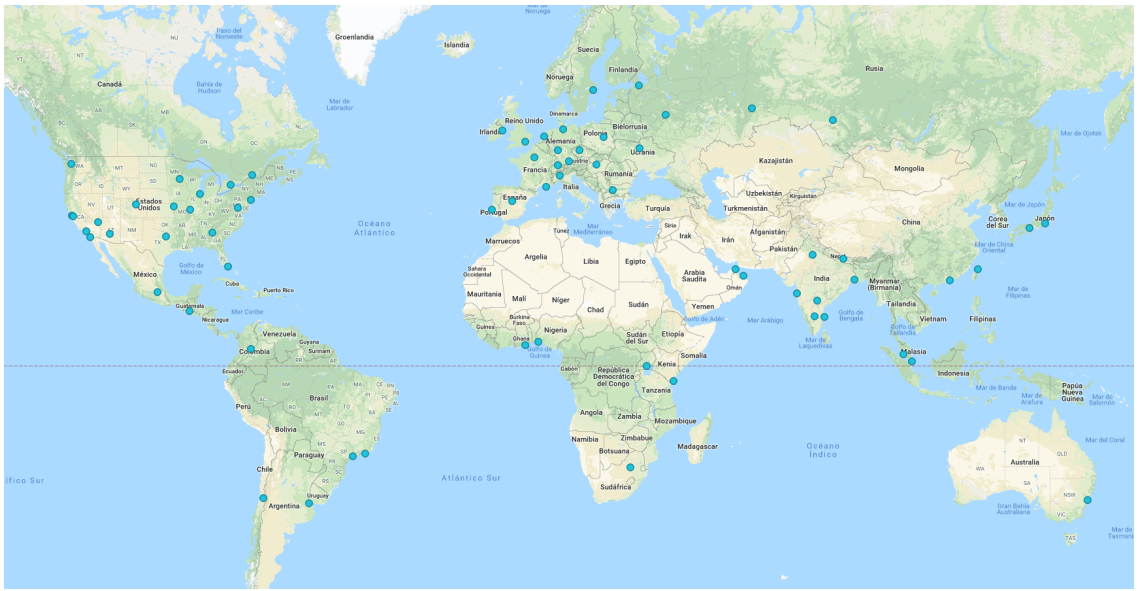

Figura 2.8: Mapa con las ubicaciones de los Points of Presence de Google.

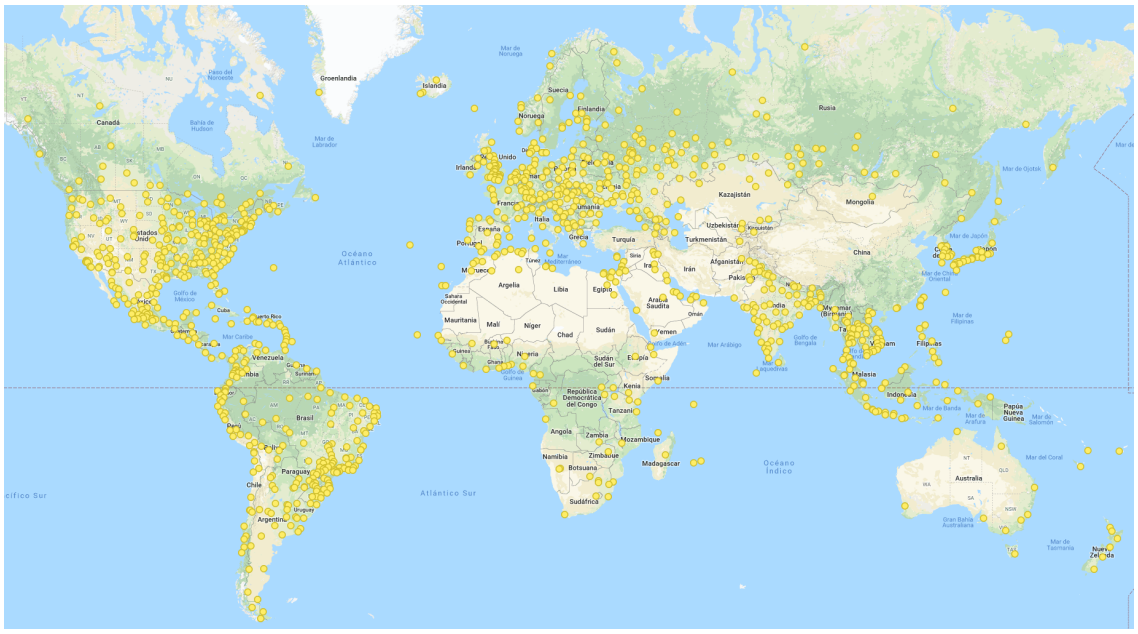

Figura 2.9: Mapa con las ubicaciones de los GGC de Google. Existen cachés que se encuentran en Alaska, en el este de Rusia y en el Océano Pacífico que no lograron salir en la imagen.

Capítulo 3

Experimentación

Se plantea realizar un experimento similar a los realizados en la sección anterior por las investigaciones previas. Se investigan las principales diferencias de contexto para encontrar posibles dificultades. Lo principal es que, con el paso del tiempo, YouTube tuvo cambios. Un ejemplo que se puede observar a simple vista, es que las conexiones a YouTube están cifradas utilizando el protocolo TLS en la capa de transporte, por lo que el capturar tráfico desde un ISP (en caso de que existiera la posibilidad), no produciría resultados útiles. Otro ejemplo, es que antes los videos eran reproducidos utilizando un plugin flash, mientras que ahora se reproducen utilizando la etiqueta de video introducida en HTML5.

3.1. Geolocalización

Antes de realizar el experimento, es necesario decidir que técnica de geolocalización utilizar. Además, se planea que la misma se realice en tiempo real, al momento que se descubren nuevos servidores.

RIPE Atlas posee una API que brinda varias funcionalidades. En particular, permite obtener probes a partir de unas coordenadas geográficas y crear medidas, tal como se puede hacer en su sitio web. Esto permite pedir probes que se encuentren cercanas a determinado lugar, enviándole coordenadas y un radio. Cabe destacar que, para cada probe, RIPE da su ubicación geográfica aproximada. En todos los casos que fue posible, se utilizaron anchors, debido a su mejor capacidad de cómputo y mayor ancho de banda.

Para seleccionar un método de geolocalización, se comienza por implementar un programa que navega por YouTube, cuya única función es recolectar dominios de servidores de contenido. De esta manera, se logra obtener un conjunto inicial de aproximadamente 650 servidores, que serán utilizados para probar el método de geolocalización utilizado. El algoritmo que se decidió utilizar para geolocalizar consiste en obtener una probe de RIPE con un RTT bajo hacia el objetivo, esto es, menor o igual a 10 ms, y luego se le asigna al objetivo la misma ubicación que la de esa probe.

El principal problema entonces, pasa por encontrar la probe con el menor ping hacia el ob-

jetivo. En los artículos analizados en el capítulo anterior, no se contempla el problema de la selección de los equipos desde los cuales realizar las mediciones, por lo que los mismos no son de ayuda. Como consecuencia, se decide generar un conjunto de ubicaciones desde las cuales se van a pedir probes (utilizando la API). Posteriormente, se generarán mediciones de ping desde las probes obtenidas hacia el objetivo a geolocalizar, asumiendo que el mismo se encuentra en alguna de esas ubicaciones. Si bien las ubicaciones son fijas, las probes pueden variar entre las distintas mediciones dependiendo de que tan frecuente se renueve el conjunto de probes.

Los 650 servidores obtenidos serán utilizados para generar el conjunto de las ubicaciones, de la siguiente manera: se decide empezar eligiendo probes que se encuentren en distintos lugares del mundo, a nivel de continentes, y se intentan geolocalizar los servidores. Esta primera aproximación da un resultado bastante importante, no hay violaciones a la velocidad de luz, lo que indica que los nombres de dominio no son anycast. Sin embargo, los RTT más bajos obtenidos se encuentran lejos del rango esperado.

Luego de esto, se busca encontrar probes con ping más bajo hacia los servidores, a partir de la información obtenida. Dado un servidor que se intentó geolocalizar, se toma la probe con el ping mínimo hacia él y se buscan probes cercanas a ella, con las que se repiten las mediciones de geolocalización. Este proceso se repite iterativamente hasta encontrar una probe para cada servidor con un ping menor a 10 ms. Realizadas repetidas iteraciones, fue posible en la mayoría de los casos. Existieron algunas excepciones que alcanzaron los 15 ms. Finalmente, se tomaron las ubicaciones de las probes resultantes y se guardaron sus coordenadas.

Este conjunto de coordenadas obtenido es entonces, el que se utilizará más adelante para geolocalizar los servidores encontrados en el experimento principal del proyecto. Es importante resaltar que es posible que se detecten casos de RTT muy altos, lo que indicaría que el objetivo no se encuentre en ninguna de las ubicaciones detectadas hasta el momento y sea necesario encontrar nuevas ubicaciones, en cuyo caso se deberá proceder de manera análoga a lo explicado anteriormente.

Se puede observar que este algoritmo es una forma simplificada de GeoPing, en la que, en lugar de utilizar un vector de retardos, se utiliza un único retardo.

Las ubicaciones obtenidas son las siguientes. En América del sur, se toman probes en Montevideo, Buenos Aires, San Pablo y Santiago. En América del Norte, Miami, Atlanta, Dallas, Denver, Los Ángeles, San José (California), Seattle, Kansas, San Luis (Missouri), Washington, Chicago, Montreal, Toronto y Nueva York. Finalmente, en Europa, Madrid, Dublin, Londres, París, Marsella (Francia), Milán (Italia), Ámsterdam, Zúrich (Suiza), Budapest (Hungría), Sofía (Bulgaria), Varsovia, Estocolmo, Frankfurt, Múnich, Kiev (Ucrania), Hamburgo y Bratislava (Hungría). Estas ubicaciones se corresponden con las ubicaciones dadas por Google en su sitio web, que fue mencionado en la sección de antecedentes. En vista de esta información, puede considerarse que la investigación tuvo un resultado satisfactorio. Se decidió utilizar 4

equipos por ubicación encontrada a la hora de realizar el experimento.

En el Cuadro 3.1 se presenta un pseudo-código del algoritmo de geolocalización. En el experimento final, el conjunto de probes utlizadas se actualizaba cada 45 minutos. En el Apéndice A se encuentra una versión más completa del algoritmo.

```
geolocalizar(objetivo) {
   // ubicaciones contiene los lugares mencionados anteriormente
   probes = obtener_probes(ubicaciones);

   medida = API_RIPE.crear_medida(probes, objetivo, tipo='ping')
   // esperar a que se complete
   resultados = API_RIPE.get_resultados(medida);

   min_ping, min_probe = procesar_resultados(resultados);

   // obtiene los datos de la probe a partir de su id
   probe = get_probe(min_probe);

   distancia = calcular_distancia(min_ping, probe);

   guardar(distancia, min_ping, probe);
```

Cuadro 3.1: Pseudo-código del proceso de geolocalización utilizado.

Antes de llegar a este algoritmo, se implementó e intentó utilizar CBG. Sin embargo, se detectaron fallas que derivaron en el algoritmo finalmente utilizado. La principal falla, era que, en muchos de los casos, la ubicación resultante del objetivo era la misma que la ubicación de la probe con el ping más bajo. Esto se debió a que se encontraron RTTs tan bajos que la intersección de todas las circunferencias daba como resultado la circunferencia de la probe con el menor ping.

Otras de las causas por las que CBG puede no ser una buena opción, es que la calibración da como resultado el estado de la red entre las distintas probes, sin embargo, los paquetes hacia los servidores de contenido entran a la red privada de Google, que probablemente funcione mejor que las conexiones entre las probes. Además, el método utilizado presenta otra ventaja, dado que no es necesario realizar mediciones periódicas entre las probes, se utilizan menos créditos de RIPE Atlas y da más flexibilidad a la hora de seleccionar las probes, las mismas pueden cambiar de una medición a la otra sin la demora que podría ser ocasionada por la necesidad de realizar medidas de calibración entre ellas.

Otro hallazgo en el proceso de selección del método de geolocalización, fue encontrarse con que los dominios de los servidores de contenido cambiaron con respecto a los mencionados en

[4]. Ahora no contienen más códigos de aeropuertos que permitan asociarlos fácilmente a un lugar. Todos terminan en *.googlevideo.com*, es decir, pertenecen todos a un mismo namespace. Además, tienen una cierta estructura: *rXX—base.googlevideo.com*, donde XX es un número y base es un string. Por ejemplo: **r7—sn-5ouxa-h8qe7.googlevideo.com** o **r4—sn-5ouxa-h8qr.googlevideo.com**, donde **sn-5ouxa-h8qr** y **sn-5ouxa-h8qe7** son las bases. Dada una base, existen varios servidores asociados a él, diferenciados por los números en el comienzo del dominio. Se plantea la posibilidad de que todos los servidores asociados a una misma base se encuentren en la misma ubicación.

3.2. Recolección de datos

Como fue mencionado anteriormente, el objetivo del experimento es recolectar datos al reproducir videos de YouTube con el fin de analizar el tráfico. La navegación se va a realizar de manera distribuida desde varios equipos, ejecutando un programa que fue realizado con el objetivo de cumplir esta tarea automáticamente. Estos equipos se encuentran ubicados en Uruguay.

El programa mencionado anteriormente, es una aplicación realizada en Java que funciona de la siguiente manera. Al ejecutarlo, carga un video inicial, ingresado como entrada, y, luego de haber pasado 15 segundos desde que se completó la carga del sitio, hace click en el próximo video a reproducir recomendado por YouTube, y así sucesivamente. Mientras tanto, se captura el tráfico y otra información sobre el video, que se envía a un servidor, encargado de procesar la información generada por estos clientes.

Para la navegación automática se utilizó el software Selenium, que está pensado para realizar pruebas automatizadas de sitios web. Este fue programado para simular la interacción entre el navegador y el usuario.

Para lograr capturar el tráfico de las reproducciones, se utilizó BrowserMob Proxy, un proxy que permite capturar el tráfico, que se integra muy bien con Selenium. Las capturas de tráfico del proxy se generan en formato HAR (HTTP Archive) [13]. Este formato permite guardar todos los mensajes HTTP entre el navegador y un sitio web, junto con los tiempos en los que fueron enviadas las peticiones, tiempo de espera y tiempo en recibir la respuesta. Poseen la ventaja de que se guardan en formato JSON, lo que permite su fácil manipulación desde cualquier lenguaje de programación. El hecho de que el tráfico esté encriptado a nivel de capa de transporte no es un problema, debido a que la captura se realiza después de que el mismo es desencriptado por el navegador.

Para extraer la información se utilizó la interfaz que brinda Selenium para seleccionar elementos utilizando selectores CSS, de los cuáles luego se obtenía y procesaba el texto adentro de los distintos elementos seleccionados. Para obtener algunos de los atributos, se utilizó información brindada en las *Estadísticas para nerds*, que es un cuadro con información del video que se encuentra reproduciéndose. Para verlo fue necesario hacer un click con el botón

derecho, utilizando una interfaz que brinda Selenium, dentro del elemento en el que se encuentra el video.

También hubo atributos que se eligió obtener utilizando la API de YouTube, dado que permite leer algunos valores de forma más fácil. Por ejemplo, la cantidad de me gusta. Esta información aparece abreviada utilizando k para miles de comentarios y m para millones, en el navegador, mientras que, en la API, aparece el número entero completo.

En el Cuadro 3.2 se presenta un pseudo-código muy reducido del programa cliente, encargado de navegar. Se puede encontrar una versión completa en el Apéndice B.

```
navegar(seed) {
   // configurar browsermob y selenium
   navegador = inicializar_navegador();
   navegador.newHar();
   navegador.get(seed);

   while(true) {
      // contiene los datos a guardar
      video = new Video();

      open_stats_for_nerds();
      video.stats = get_stats_for_nerds_info();
      // obtener id del video
      video.id = get_id_video();
      video.info_api = API_YOUTUBE.get_info_video(video.id);
      video.HAR = navegador.getHar();

      enviar_video_al_servidor(video);

      proximo = navegador.getProximoVideo();
      navegador.resetHar();
      proximo.click();
   }
}
```

Cuadro 3.2: Pseudo-código programa cliente encargado de navegar por el sitio YouTube recolectando datos.

El servidor antes mencionado, es una aplicación realizada en Python, que se encarga de atender peticiones que contienen datos obtenidos al reproducir un video de YouTube, almacenando la información recibida en una base de datos, y, analizando los servidores de

contenido de esa reproducción. En caso de detectar un servidor de contenido nuevo, se inicia un proceso de geolocalización del mismo.

Las capturas de tráfico HAR se almacenaron en archivos de texto, mientras que los otros datos obtenidos de los videos se almacenaron en una base de datos relacional, en una tabla donde cada reproducción de un video se identifica por un número auto incremental, que también es utilizado para obtener el archivo HAR correspondiente a esa reproducción. En la base de datos se cuenta con dos tablas más. Una tabla que guarda los servidores de contenido asociados a cada video y una tabla encargada de guardar los datos resultantes del proceso de geolocalización.

En el Apéndice C se encuentra un pseudo-código que enseña el funcionamiento del servidor.

Se utilizaron 6 equipos para el experimento. De estos, 4 se encontraban en el edificio del InCo, conectados al AS 1797 a través del proxy de la facultad, y 2 estaban conectados a la red ADSL de Antel, en el AS 6057. De los que se encontraban en la facultad, 3 tenían sistema operativo Ubuntu y uno Windows. De los conectados por ADSL, uno tenía Ubuntu y uno Windows. En la Figura 3.1 se encuentra la arquitectura de red del experimento. Los equipos que se encontraban en la facultad se conectaban a internet a través del proxy de la facultad.

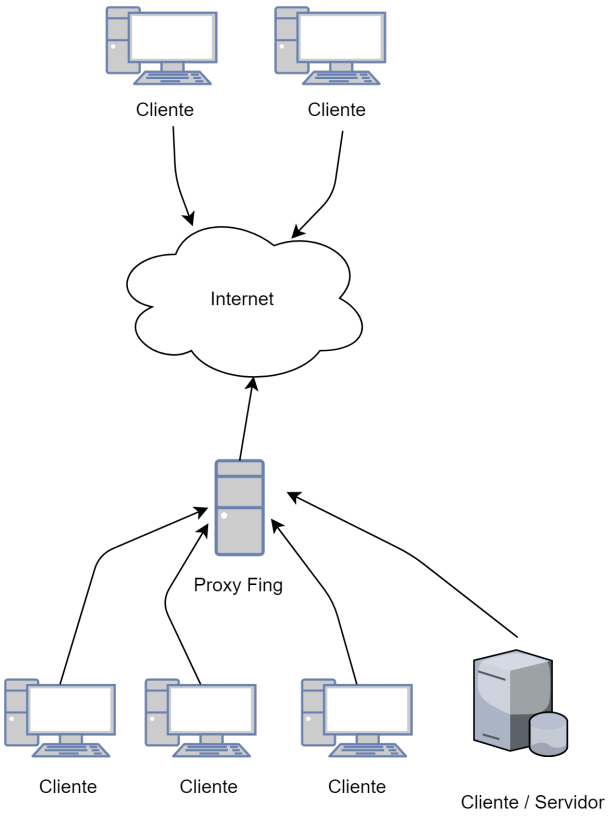

Figura 3.1: Arquitectura a nivel de red del experimento. Se observan los equipos y su forma de conectarse a internet.

La forma de ejecución fue la siguiente. Cada equipo comenzó a correr el programa cliente con un video semilla inicial y cada 24 horas el programa era reiniciado, y se utilizada otra semilla inicial. Se eligió un conjunto de videos iniciales separados en cuatro categorías. Cada categoría tiene 7 videos.

- Noticias de los Países Bajos: se obtuvieron utilizando el buscador de YouTube utilizando un proxy en ese país.
- Videos relevantes a Uruguay. Por ejemplo, noticias uruguayas y videos de fútbol de la selección.
- Tendencias: Se obtuvieron de la página inicial de YouTube.
- Noticias relevantes a Estados Unidos: Se utilizó el buscador y se obtuvieron noticias relacionadas a Trump.

Se intentaron elegir las categorías de forma que hubiera una con contenido que no fuera relevante a Uruguay, una que lo fuera, una que puede contener videos que sean relevantes a parte de la población de Uruguay y finalmente, videos sugeridos por YouTube, populares.

En cada equipo ubicado en la facultad se utilizaron semillas de una categoría distinta. En los equipos conectados por ADSL, se utilizaron las semillas de tendencias y noticias de los Países Bajos.

El experimento se realizó durante un período de un poco más de una semana, entre las 17:48 hs del 22/9/2018 y las 15:41 hs del 30/9/2018. Se utiliza una semilla al día para cada equipo.

Debido a limitaciones de la cantidad de medidas permitidas por RIPE Atlas al día, no todos los servidores se pudieron geolocalizar al mismo tiempo en el que fueron descubiertos. Esto sucedió sobre todo en los primeros días del experimento, cuando una gran cantidad de los servidores detectados eran nuevos.

Capítulo 4

Resultados

En la presente sección se presentan los resultados obtenidos del experimento. Se realiza también un breve análisis de los mismos, se evalúa el algoritmo utilizado para la geolocalización y se analiza en profundidad las relaciones entre un video y sus servidores de contenido.

Se logró obtener un total de 67496 videos, de los cuales ninguno es un stream en vivo. La duración promedio de los videos es de 6,35 minutos. Hay 12278 canales distintos.

4.1. Descripción del conjunto de datos

Se va a realizar una descripción de los datos obtenidos, describiendo los atributos y las distribuciones de los mismos.

4.1.1. Atributos

Los atributos obtenidos para cada video reproducido se separaron en numéricos o categóricos, dependiendo de los valores que podían tomar. A continuación, se presenta una lista con la totalidad de los atributos.

- La categoría del video, obtenida utilizando una API de YouTube.
- La definición del video, es decir, si se reprodujo en baja o alta definición.
- Un atributo *redirection* que indica el usuario fue redirigido (no cuenta la cantidad de veces) o no.
- *origen*, representa desde cual equipo se reprodujo el video.
- *servidor_inicial*, el primer servidor al cual se le intentó pedir contenido.
- *next_video* identificador del próximo video a reproducir según la reproducción automática de YouTube.
- *video_id* identificador del video asignado por YouTube.

- *s_principal* y *s_secundario*, servidores asignados por YouTube a los cuales se debe pedir el contenido.
- *first_byte_server* y *time_to_first_byte* representan el primer servidor que envió contenido multimedia de ese video y el tiempo desde que se hizo la solicitud a YouTube para reproducir ese video hasta que llegó el primer byte de contenido.
- *stats_connection* velocidad en Kbps de la conexión hacia uno de sus servidores de contenido.
- *likes, dislikes, views, commentcount*. Cantidad de me gusta, no me gusta, visitas y comentarios.
- *duracion*, duración en segundos del video.
- *idioma*, idioma del video.
- *channel*, identificador del canal que subió el video a YouTube.
- *fecha_upload*, fecha en la que el video fue subido a YouTube.
- *titulo*, título del video.
- *horario*, horario en el que se reprodujo el video.
- *Servidores de contenido*, conjunto servidores que enviaron contenido multimedia para la reproducción.

Además, se cuenta con las capturas de tráfico en formato HAR para cada una de las reproducciones, las cuales fueron utilizadas para obtener algunos de los atributos mencionados anteriormente.

Para facilitar el estudio del horario, se realizaron franjas horarias y cada video fue asignado a una franja según la hora en la que se reprodujo. Entre la 1 y las 7 de la mañana se lo asignó a la categoría 'madrugada', entre las 7 y las 14 horas, 'mañana', de las 14 a las 17 'tarde', entre las 17 y las 21 'noche' y finalmente entre las 21 y la 1 'noche_tarde'. Se intentó elegir franjas en las que se pensara que iba a haber una cantidad similar de personas mirando videos durante todo el horario.

La cantidad de videos que se consumieron en alta definición fue del 86 % y en baja definición del 14 %.

La cantidad de videos en los que el usuario fue redireccionado por lo menos una vez, es de aproximadamente un 20 %.

4.1.2. Análisis de los atributos categóricos

Se expone el análisis de los atributos categóricos considerados los más relevantes para describir el conjunto de datos.

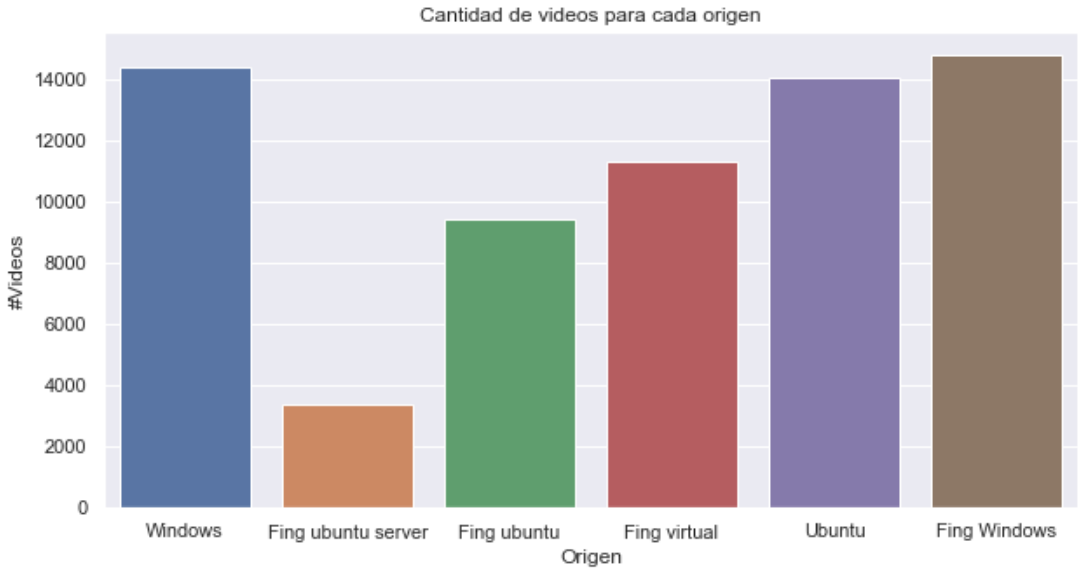

Figura 4.1: Cantidad de videos para cada equipo participante del experimento.

La Figura 4.1 evidencia una cantidad dispareja de videos para cada origen. La cantidad de videos por origen depende de la velocidad de la conexión de cada equipo, y la velocidad que demoraba en cargar el sitio cuando se cambiaba de video. Esto último depende además de la velocidad de la conexión, del hardware del equipo.

La gran diferencia de cantidad de videos entre el equipo servidor y el resto se debe a que, por problemas en ese equipo, el cliente dejó de funcionar durante el tercer día del experimento. Se optó por no reiniciarlo, dado que ese equipo era más importante como servidor que como cliente.

El Cuadro 4.1 presenta las categorías junto con el porcentaje de videos que pertenecen a cada una. Se evidencia la presencia de categorías más visitadas que otras. Las categorías en negrita significan que algún video dentro de las semillas iniciales se encontraba en esa categoría. Se puede ver que existen categorías que, aunque haya semillas iniciales en esa categoría, no hay una gran cantidad de videos de esa categoría. También se observa la aparición de algunas categorías nuevas, de las cuales algunas poseen un gran porcentaje de la cantidad de videos reproducidos. Esto muestra que existen otros factores además de la categoría de un video en el algoritmo de recomendación de YouTube.

Se consideró útil para el experimento tener asignado un idioma a cada video. Esto se pudo lograr utilizando la API de Google Translate, utilizando como entrada el título del video. La

Categoría	**% Videos**
Entretenimiento (24)	30.351 %
Personas y Blogs (22)	15.370 %
Música (10)	14.506 %
Juegos (20)	8.340 %
Películas y Animación (1)	8.172 %
Educación (27)	7.372 %
Deportes (17)	4.812 %
Noticias y Política (25)	2.407 %
Comedia (23)	2.352 %
Activismo (29)	1.527 %
Ciencia y Tecnología (28)	1.320 %
Vehículos (2)	1.287 %
Mascotas y Animales (15)	0.930 %
Viajes y Eventos (19)	0.612 %
Howto & Style (26)	0.446 %
Shows (43)	0.187 %
Trailers (44)	0.006 %

Cuadro 4.1: Porcentaje de videos en cada categoría. El número en paréntesis es el identificador de la categoría utilizado por la API de YouTube.

Figura 4.2, un gráfico con la cantidad de videos para cada idioma. El idioma más observado es el inglés, seguido del español, portugués, alemán y holandés. Los idiomas que tenían menos de 1600 videos asignados se agruparon en el conjunto llamado Otros. Entre ellos se encuentran árabe, rumano, chino, polaco, turco, coreano, italiano, francés, japonés, ruso, etc.

4.1.3. Análisis de los servidores localizados

Se realiza un breve análisis del conjunto de servidores localizados, estudiando el error de la geolocalización y su ubicación.

Corresponde comenzar por analizar el error del algoritmo de geolocalización en la práctica. El RTT mínimo más alto que posee un servidor de los localizados es 14.83 ms. Asumiendo que todos los retardos son despreciables, excepto el de propagación, es posible calcular la distancia que pudo haber viajado el datagrama, utilizando la siguiente fórmula:

$$distancia \ = \ \frac{RTT}{2} * \frac{4}{9} * C$$

El valor C se corresponde con la velocidad de la luz en el vacío. El coeficiente $\frac{4}{9}$ es el mismo que se utiliza en el artículo [14] para geolocalización. Recordando de los antecedentes, los autores de CBG utilizaban $\frac{2}{3}$. Se prefirió seleccionar el primer coeficiente debido a que el

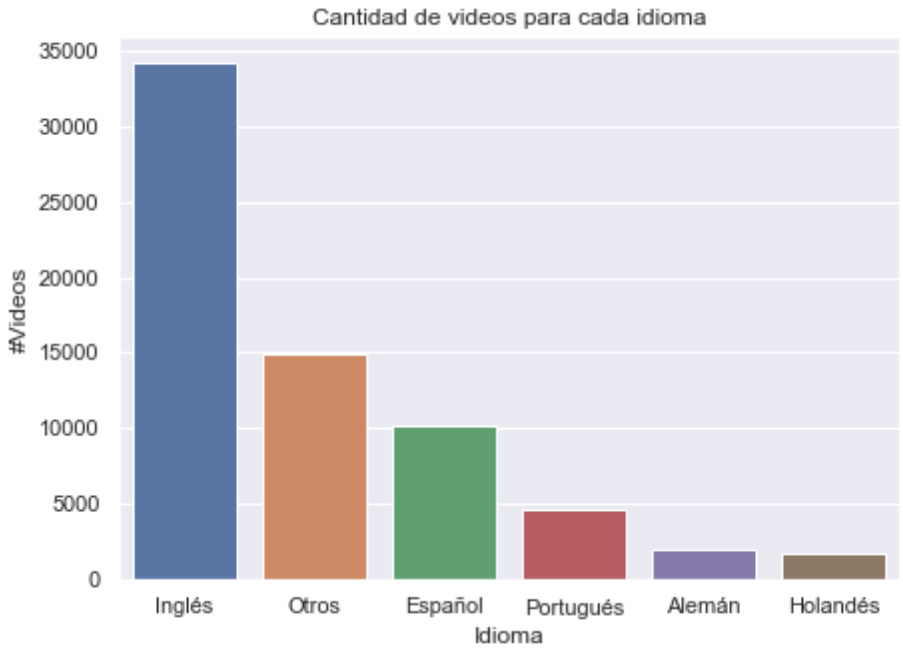

Figura 4.2: Cantidad de videos para cada idioma.

artículo [14] es más reciente que el de CBG.

Si se sustituye el RTT por 14.83 ms, se obtiene que la distancia máxima que pudo haber viajado el datagrama es de 989 km. Es decir, el error máximo en la geolocalización es de 989 km.

Un detalle a tener en cuenta, es que en este cálculo se asume que el tiempo que demora el paquete en ir desde el equipo al objetivo y desde el objetivo al equipo es el mismo, algo no necesariamente correcto.

Para cada servidor, se guardaron sus coordenadas, la distancia máxima a la que se puede encontrar de esas coordenadas, el ping mínimo encontrado desde una probe de RIPE, el identificador de la probe con el ping mínimo, el país donde se encuentra la probe y el identificador de la medida de RIPE utilizada para la geolocalización.

Observando el histograma de la Figura 4.3, se puede ver que el 75 % de los servidores tuvieron un error menor a 100 km. Se considera este resultado bastante bueno, y, además, muestra la gran utilidad de la plataforma Atlas. Sin embargo, estos resultados dependen de que las ubicaciones de las probes de Atlas sean correctas. En su página de preguntas frecuentes, Atlas informa que las ubicaciones se encuentran ofuscadas una distancia de un máximo de 1 km de la ubicación real. No obstante, nada asegura que la ubicación considerada como real por Atlas sea correcta.

Se identifican dos tipos de peticiones a los servidores de contenido. Unas que efectivamente

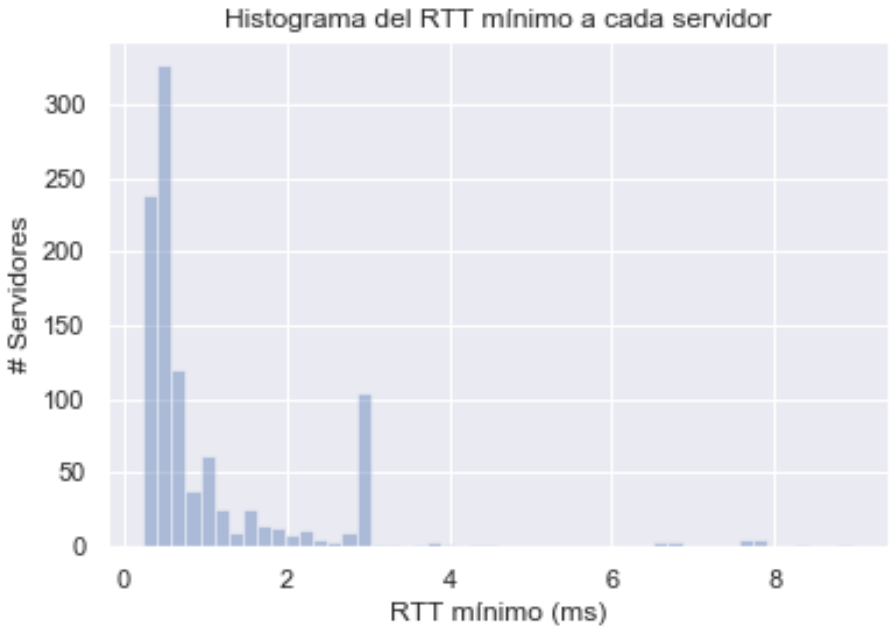

Figura 4.3: Histograma del RTT mínimo. Se observa que la mayoría de los servidores tiene un RTT mínimo muy pequeño, mostrando un error en la geolocalización bastante pequeño en la mayoría de los casos e indicando que puede haber probes de RIPE en la misma red que los servidores de Google.

contienen contenido multimedia y otras que no lo hacen. Las primeras se pueden identificar debido a que el recurso que piden en los servidores es **/videoplayback** y el cabezal HTTP que indica el contenido de la respuesta es video o audio. En el segundo caso, el recurso a pedir es **/videogoodput** y el contenido de la respuesta es de tipo *application/octet-stream*. Se observa que existen servidores a los que se enviaron los dos tipos de peticiones. A partir de ahora, el estudio se va a enfocar en los servidores que recibieron peticiones de contenido multimedia o, dicho de otra manera, los que se utilizaron para obtener contenido.

En el Cuadro 4.2 se puede apreciar cuantos servidores de contenido se encuentran en cada país. En Estados Unidos con 731 servidores, es donde se encontró la mayor cantidad, superando por 6 veces la cantidad del segundo lugar, Brasil, con 116.

La Figura 4.4 muestra un mapa con los lugares donde se encontraron servidores. Si bien hay más de 1000 servidores, en el mapa se puede apreciar que se concentran en tan solo 16 ubicaciones. Si se compara con la Figura 2.2, que muestra las ubicaciones que habían sido encontradas en el 2012 por el artículo [4], se aprecia que muchas de las ubicaciones actuales parecen ser las mismas que las halladas en ese momento.

Se descubrió que, al reproducir un video, en la mayoría de los casos se envían dos direcciones URL de servidores de contenido, a diferencia de lo mencionado en la sección de antecedentes, donde se dijo que se enviaba una sola. Una de las direcciones corresponde a un servidor

País	# Servidores
Estados Unidos (US)	731
Brasil (BR)	116
Países Bajos (NL)	49
Argentina (AR)	42
Uruguay (UY)	42
Francia (FR)	34
Chile (CL)	12
Canadá (CA)	8
Italia (IT)	6

Cuadro 4.2: Cantidad de servidores de contenido en cada país.

Figura 4.4: Mapa con las ubicaciones de los servidores de contenido de Google geolocalizados.

de contenido primario, del cual siempre se intenta pedir el contenido. La dirección del otro servidor, corresponde con un servidor secundario, el cual es utilizado si hay problemas de conectividad al primario. Se observa que en caso de no recibir un servidor secundario y ocurrir problemas de conectividad, se contacta con el servidor **redirector.googlevideo.com**, el cual envía un dominio de un nuevo servidor de contenido.

Estos servidores se pueden obtener analizando la URL de una petición a un servidor de contenido, observando el parámetro de la consulta *mn*. Por ejemplo, si en una petición al servidor de contenido *r3—sn-5ouxa-h8qk.googlevideo.com* se encuentra *mn=sn-5ouxa-h8qk %2Csn-x1x7zn7s*, significa que el servidor de contenido primario es *r3—sn-5ouxa-h8qk.googlevideo.com* y el servidor secundario es *r3—sn-x1x7zn7s.googlevideo.com*.

Estudiando sus ubicaciones, en el caso de los servidores primarios, la totalidad de ellos se encuentran en Uruguay. Para el caso de los servidores secundarios, la mayoría de ellos están

localizados en países de la región o en Estados Unidos. Sin embargo, existe una pequeña cantidad de videos, cuyos servidores secundarios se encuentran en Francia, Canadá e Italia, lugares más alejados. La Figura 4.5 muestra cuantos videos poseen su servidor secundario en cada país.

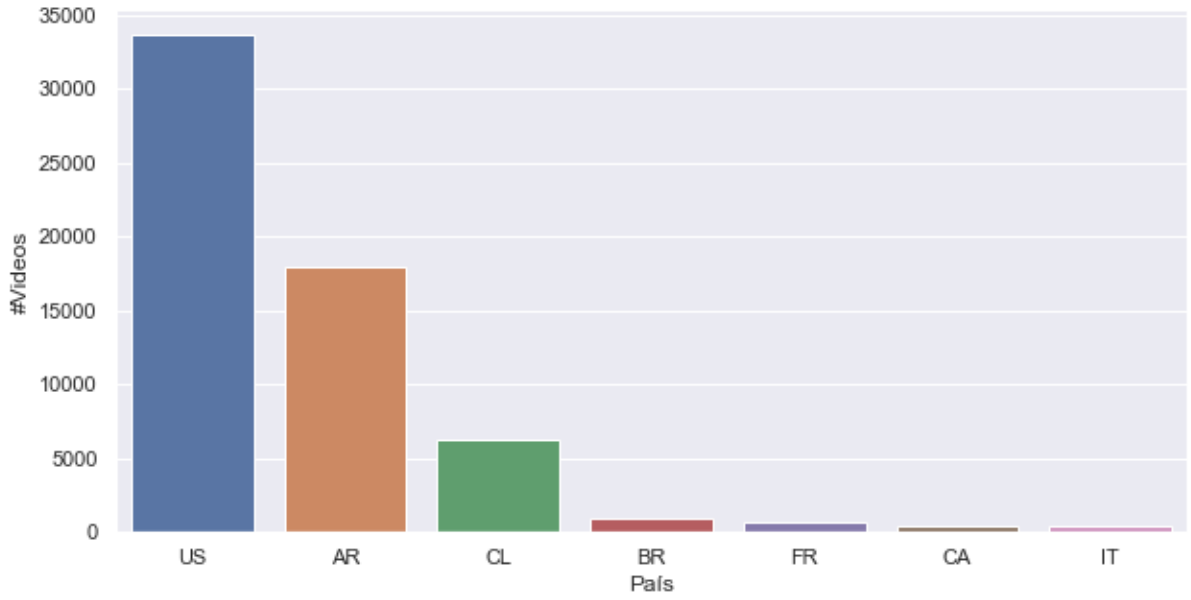

Figura 4.5: Ubicación de los servidores secundarios de los videos.

Se busca una relación entre el idioma de un video y su servidor secundario. Los resultados arrojan información suficiente para concluir que dicha relación parece no existir. En todos los casos, los idiomas con más cantidad de videos eran el español y el inglés. En Brasil se encontraron videos en portugués, pero en muy baja cantidad en comparación con la cantidad obtenida en los servidores de Estados Unidos, Chile y Argentina. En Francia no se detectaron videos en francés, aunque si había en Canadá. De los videos en italiano, solo dos se encontraban en servidores italianos.

4.2. Agrupación de dominios

En la sección anterior se planteó la posibilidad de que todos los servidores con una misma base se encuentren en el mismo lugar. Con el objetivo de investigar este propósito, se analizan las direcciones IP de los servidores. Se descubre que todas las direcciones asociadas una misma base son continuas, es decir, si la dirección para r1—base1.googlevideo.com la dirección es xx.yy.zz.01, la dirección para r2—base1.googlevideo.com es xx.yy.zz.02. Se busca cual es el prefijo más largo tal que para cada base, todas las direcciones de esa base se puedan agrupar bajo ese prefijo. Se obtiene que dicho prefijo es /26. Es decir, para cada base todas sus direcciones IP caen dentro de un prefijo /26, no obstante, dentro de ese prefijo hay direcciones

no asociadas a un servidor de contenido, o puede haber servidores no detectados dentro del experimento.

Se toman todos los servidores asociados a una misma base y se mira su ubicación resultante del algoritmo de geolocalización. Existen casos en los que todos los servidores de una misma base se encuentran en la misma ubicación. En el caso de las bases con servidores en distintos lugares, se calcula la distancia geográfica entre todos los pares de servidores y se considera el error en la geolocalización. En caso de que la distancia entre dos equipos sea mayor a la suma de los errores en la geolocalización, los equipos no pueden encontrarse en el mismo lugar. En el caso contrario, es posible que sí. Debido a que la distancia fue menor a la suma de los errores en todos los casos, **se puede asumir que los servidores con misma base se encuentran todos en el mismo lugar.**

4.3. Grafo de Redirecciones

Analizando las capturas, se investigan los casos en los que hay más de un servidor de contenido. Se encuentran dos posibilidades, redirecciones y problemas de conectividad que hacen que el navegador intente contactar al servidor secundario. Los mecanismos de redirección son pedidos HTTP con código de estado de redirección, con un nuevo servidor o mensajes con tipo de contenido text/plain, y un nuevo servidor en el cuerpo del mensaje. Los casos de los problemas de conectividad se identifican mirando los tiempos de respuesta de las peticiones.

La Figura 4.6 muestra la cantidad que redirecciones que hubo. Se observa que, para la amplia mayoría de los videos no ocurrieron redirecciones. Dentro de los videos redirigidos, lo más común son 1 o 2 redirecciones, existiendo una pequeña cantidad de videos que son redirigidos 3 veces y muy pocos casos extremos de videos que son redirigidos entre 4 y 6 veces.

Para estudiar las relaciones entre los distintos servidores, se construyó un grafo dirigido, $G = (V, A)$, dónde V pertenece al conjunto de servidores que participaron en una redirección y (a, b) pertenece a A si ocurrió una redirección desde el servidor a al servidor b. Dado que los servidores con misma base en su nombre de dominio se encuentran en el mismo lugar físico, para facilitar el estudio se decidió agrupar dichos servidores en un nodo solo. De esta manera, se obtiene un grafo de 226 nodos y 2374 aristas.

4.3.1. Jerarquía

Se estudia el grafo para determinar si continúa existiendo una jerarquía de servidores (como mostraban las investigaciones previas). Lo primero a notar es que solamente existen ciclos de largo 1. Esto significa que existen redirecciones entre servidores con misma base y distinto número, que, al agruparlos, en el grafo se ve reflejado como ciclos de largo 1. Una vez eliminados estos ciclos, si se genera el grafo no dirigido resultante de eliminar las direcciones en las aristas, se observa que posee una sola componente conexa y que se forman ciclos. Por lo tanto, el grafo no es un árbol.

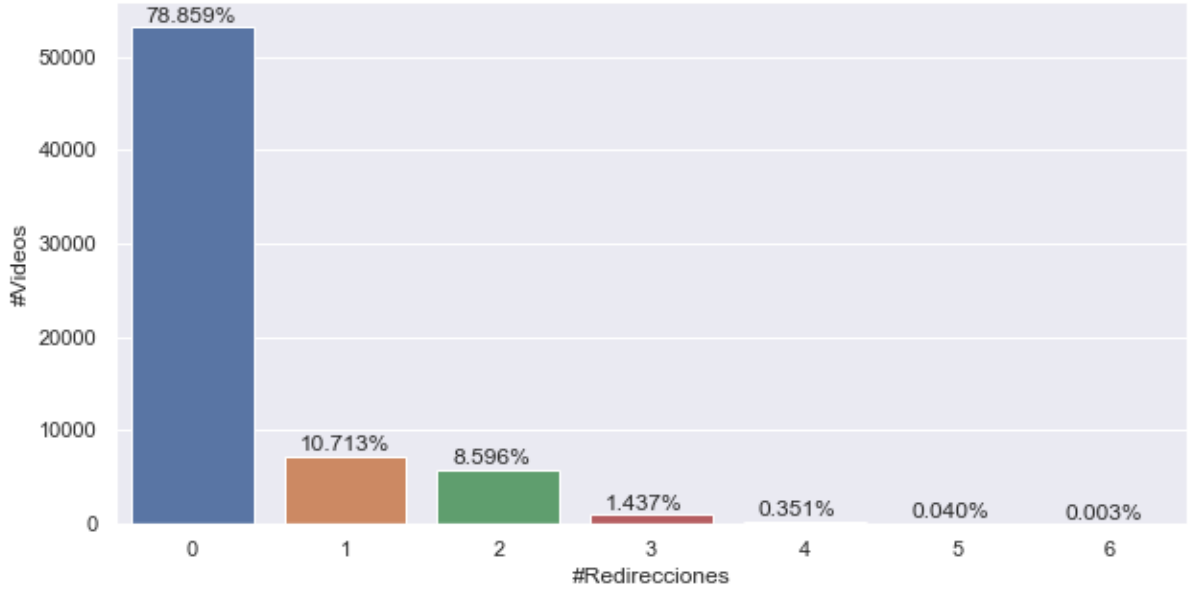

Figura 4.6: Cantidad de redirecciones por cantidad de videos.

El nivel de jerarquía de un servidor se establece analizando que servidores redirigen a él. Aplicado al grafo, esto es estudiando sus antecesores. Se establecen como los nodos del primer nivel de la jerarquía a los servidores iniciales, es decir, los que tienen grado de entrada cero. Los servidores del segundo nivel son los que fueron redirigidos por alguien en el primer nivel de la jerarquía. Continuando de esta manera, se detecta un nivel más de jerarquía, dando como resultado una jerarquía de tres niveles. Se considera al primer nivel como el de jerarquía más baja.

Al agrupar todos los nodos de cada nivel de la jerarquía en uno solo, se obtiene un grafo dirigido de tres nodos y dos aristas, mostrando que no hay saltos entre jerarquías. Existen aristas entre nodos de la misma jerarquía. Se concluye que, dado un servidor, solamente pudo haber una redirección hacia él desde un servidor de jerarquía más baja o un servidor en el mismo nivel de jerarquía. Más precisamente, las redirecciones entre servidores del mismo nivel de jerarquía son las que generan ciclos al eliminar las direcciones en las aristas, evitando que se forme un árbol. En la Figura 4.7 se muestra una gráfica con la cantidad de servidores en cada nivel de la jerarquía.

4.3.2. Análisis de la jerarquía por país

Se investiga ahora sobre la ubicación de los servidores en los distintos niveles de la jerarquía. Los servidores del primer nivel se encuentran todos en Uruguay. Esto es coherente con el hecho de la existencia de cachés en Uruguay. Los del segundo nivel se encuentran en Estados

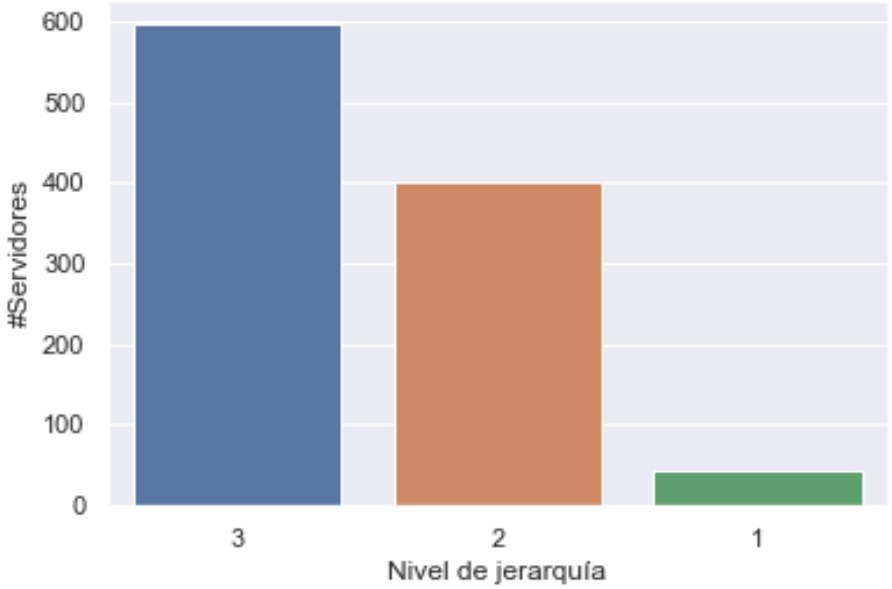

Figura 4.7: Cantidad de servidores en cada nivel de jerarquía.

Unidos, Brasil, Argentina, Francia, Chile, Canadá e Italia, tal como muestra la Figura 4.8. Finalmente, los del tercer nivel de jerarquía se encuentran en Estados Unidos y los Países Bajos, indicado en la Figura 4.9. Resulta sorprendente el hecho de que haya servidores de jerarquía dos en lugares tan alejados de Uruguay como Francia, Italia y Canadá mientras que, en el último nivel, los servidores se concentran en Estados Unidos y los Países Bajos. Se esperaría que la distancia entre el usuario y el servidor fuera aumentando con la cantidad de redirecciones, pero esto no sucede. Además, se considera que la existencia de servidores en los Países Bajos en este nivel esté relacionada con el hecho de que dentro de las semillas iniciales había un conjunto de videos relevantes a ese país, junto con la existencia de un datacenter en ese lugar.

Si se agrupan todos los nodos del mismo país en uno solo, se obtiene un grafo, enseñado en la Figura 4.10, que muestra a que país redirige cada país. Como era de esperarse, Uruguay redirige a todos los países en el segundo nivel de jerarquía. En cuanto al tercer nivel, todos redirigen a Estados Unidos y, solamente Francia e Italia redirigen a los Países Bajos. Algo sorprendente, es que existen servidores en Estados Unidos que redirigen a servidores en Argentina, y es el único lugar al que Estados Unidos redirige fuera de él. Se sospecha que esto puede suceder debido a que en Argentina se encuentra el PoP más cercano a nuestro país.

Otra observación que se puede realizar, es que, si se eliminan los nodos pertenecientes a servidores ubicados en Chile, el grafo se divide en 15 componentes conexas. Por lo tanto, Chile es un país importante en la conectividad.

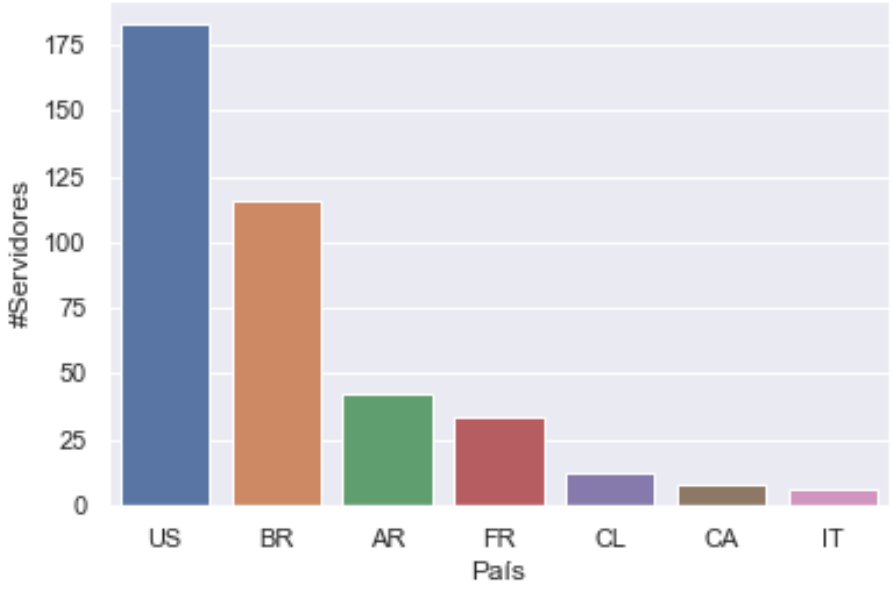

Figura 4.8: Cantidad de servidores en el nivel 2 de jerarquía ubicados en cada país.

4.3.3. Análisis de la cantidad de bytes enviados por cada país

El país del que se enviaron más bytes fue Uruguay. Este resultado podría estar sesgado por el hecho de que todos los videos se visitan por aproximadamente la misma cantidad de tiempo, y, al haber una redirección, el tiempo hasta que se empieza a llegar contenido es mayor, por lo que se reproduce menos cantidad de tiempo. La cantidad de bytes para los otros países se puede ver en el Cuadro 4.3.

Esto muestra que Google utiliza cachés en Uruguay, los cuales enviaron la mayor parte del contenido consumido en el experimento. El segundo lugar, ocupado por estados unidos, no es ninguna sorpresa, dado que la mayoría de los datacenters de Google se encuentran en ese país. Además, hay datacenters de Google tanto en Chile como en los Países Bajos. En el

País	**# Bytes**
Uruguay (UY)	5.62×10^{11}
Estados Unidos (US)	7.52×10^{10}
Argentina (AR)	1.77×10^{10}
Chile (CL)	8.22×10^{9}
Francia (FR)	1.19×10^{9}
Brasil (BR)	7.12×10^{8}
Países Bajos (NL)	3.50×10^{8}
Italia (IT)	2.25×10^{8}
Canadá (CA)	5.50×10^{7}

Cuadro 4.3: Cantidad de bytes enviados desde cada país.

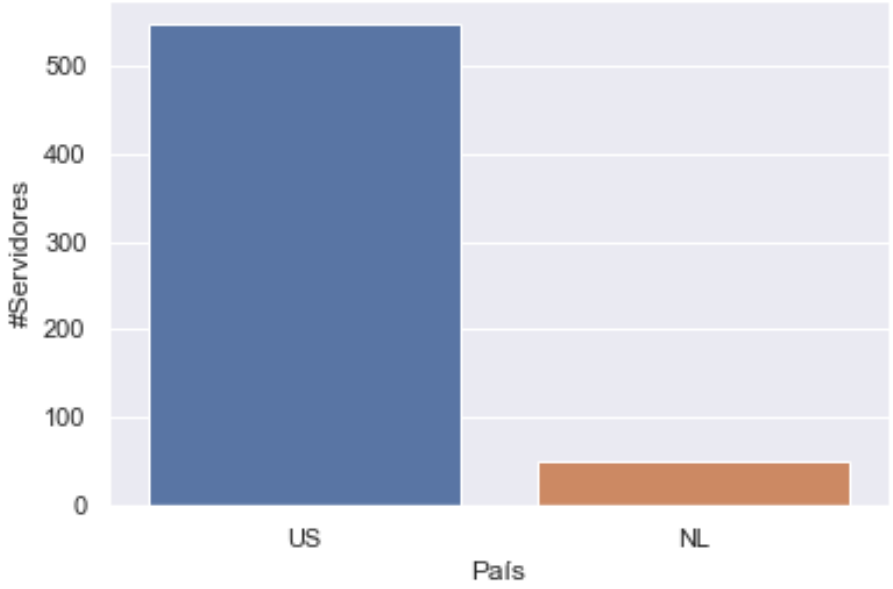

Figura 4.9: Cantidad de servidores en el nivel 3 de jerarquía ubicados en cada país.

resto de los países, se encuentra por lo menos un Edge Point of Presence de Google, junto con Google Global Caches (mencionados en la sección de antecedentes.), según indica Google en su sitio web [11]. Es posible que los servidores de los cuales se obtienen los videos pertenezcan a un GGC ubicado en otro país, conectándose con Uruguay mediante el point of presence ubicado cerca del destino y algún point of presence cercano a Uruguay, posiblemente en Chile, Argentina o Brasil. Un hecho que aporta evidencia a esta hipótesis es que los resultados de la geolocalización muestran un retardo muy bajo en la mayoría de los casos, algo que es posible solo si los equipos utilizados se encuentran en la red de Google o muy cercanos a ella. Recordando que se utilizaron (cuando era posible) probes de tipo anchor, que suelen ser instaladas por empresas grandes como ISPs, es posible que algunas se encuentren en el mismo datacenter (del ISP) que los GGC de Google.

La Figura 4.11 muestra la cantidad de bytes enviados por país y por nivel de jerarquía. Se observa que, desde Estados Unidos, se enviaron casi la misma cantidad de bytes desde el nivel dos que desde el nivel tres. Chile, a pesar de tener bastante importancia en la conectividad del grafo, se encuentra en el cuarto lugar en bytes enviados, después de Uruguay, Estados Unidos y Argentina. El resto de los países tiene poca importancia en comparación a ellos. Sin embargo, hay dos observaciones que vale la pena realizar. Llama la atención que Francia tenga más bytes enviados que Brasil, dado que es un país que se encuentra más cerca y se esperaría que tuviera una importancia similar a la de Argentina o Chile.

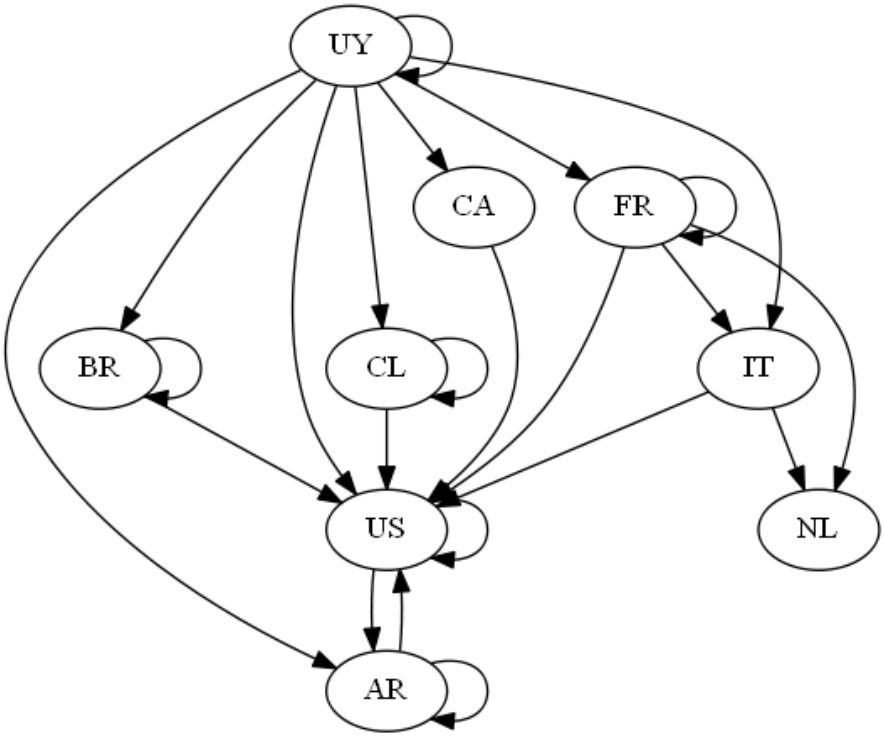

Figura 4.10: Grafo de redirecciones, se agruparon los nodos que se encuentran en el mismo país en uno solo. Una arista entre un nodo A y un nodo B significa que existe por lo menos un servidor en el país A que redirige a un servidor en el país B.

4.3.4. Relación entre el idioma de un video y sus servidores de contenido

A partir de la observación anterior, se decide estudiar para cada país, de que idiomas son los videos que sus servidores enviaron, buscando si hay alguna relación entre el idioma hablado en el país e idioma del video. Lo primero a observar el idioma más enviado por todas las ubicaciones es inglés, siendo Uruguay el que más videos envió en todos los idiomas, seguido por Estados Unidos. Si bien hasta el momento no parece haber ninguna sorpresa, debido al uso de cachés en Uruguay y la cantidad de servidores en Estados Unidos, llama la atención cuando se descubre que los servidores en Italia, Francia y los Países Bajos no enviaron videos en los idiomas hablados en esos países. Debido a la distancia, se esperaría que los servidores en esos países enviaran videos en los idiomas hablados en esos países. En el caso de Brasil, solamente se envió un video en portugués.

De este análisis se puede concluir que no hay una relación entre el idioma de un video y el hablado en el país donde se encuentran sus servidores de contenido. Además, la hipótesis realizada en la segunda observación que se encuentra al final de la sección anterior, es equivocada, dado que los videos en holandés no se enviaron desde servidores ubicados en los Países Bajos.

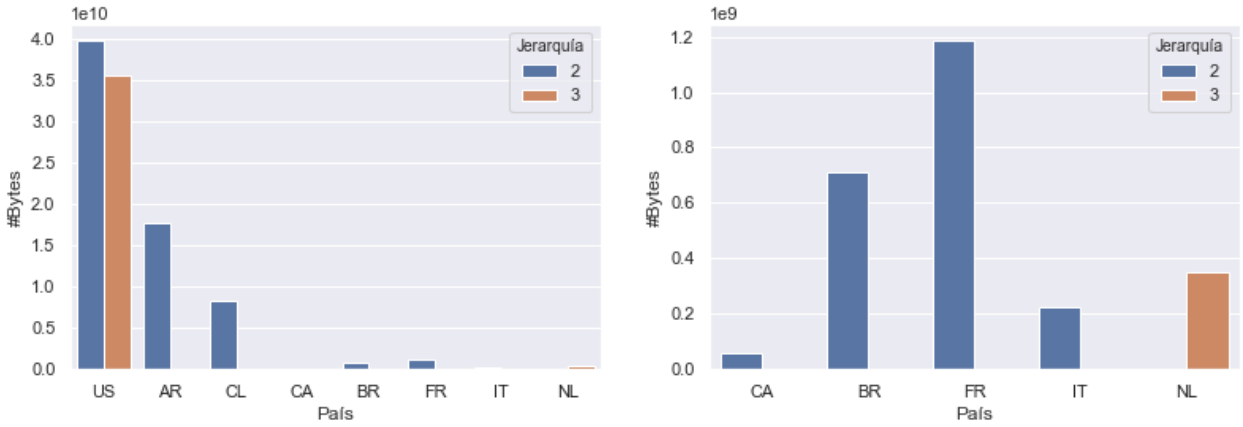

Figura 4.11: Cantidad de bytes enviados por cada país, según el nivel de jerarquía de los servidores. La gráfica de la derecha es la misma que la izquierda pero se omiten Estados Unidos, Argentina y Chile. En ambas gráficas Uruguay no se encuentra para facilitar la lectura.

4.3.5. Funcionamiento del stream

En caso de haber una redirección, ¿Ocurren al comenzar a reproducir el video o pueden suceder a la mitad?, ¿Todo el contenido restante es enviado por el nuevo servidor o vuelve a pedir al servidor anterior?, si hay más de una redirección, ¿Suceden una después de la otra o se pide contenido a servidores intermedios?, cuando hay problemas de conectividad y se contacta a un servidor secundario, ¿Se vuelve a pedirle contenido al principal? Para intentar resolver estas interrogantes, se realizaron gráficas para cada reproducción, que muestran en que tiempos se enviaron mensajes hacia cuales servidores, junto con su jerarquía y si la respuesta fue una redirección o contenido. Se seleccionaron algunas para responder estas dudas. No se encuentran las peticiones que no recibieron respuesta debido a problemas de conectividad.

La Figura 4.12 ilustra un ejemplo en el que ocurre una redirección luego de pasado bastante tiempo del comienzo de la reproducción del video. En la gráfica se observa como el video comienza a reproducirse desde el servidor inicial en el primer nivel de jerarquía y pasados los 20 segundos es redirigido a un servidor en el segundo nivel de jerarquía. Esto muestra que las redirecciones pueden ocurrir en cualquier momento de la reproducción. También puede significar que el video puede cachearse totalmente o parcialmente. Además, se vuelve a pedir contenido al servidor anterior. Al contrario, en la Figura 4.13 se observa un ejemplo opuesto, donde todas las redirecciones ocurren al comienzo, pasando por todos los niveles de la jerarquía, y no se vuelve a pedir contenido a servidores anteriores. Estos dos ejemplos sirven para contestar las primeras dos preguntas, en las que las dos alternativas planteadas para cada pregunta son posibles.

La ya mencionada Figura 4.13 junto con la Figura 4.14 brindan la respuesta para la tercera pregunta. Las redirecciones pueden ocurrir una después de la otra, o puede suceder que entre las redirecciones se envíe contenido. El primer caso es el visto en la Figura 4.13, mientras

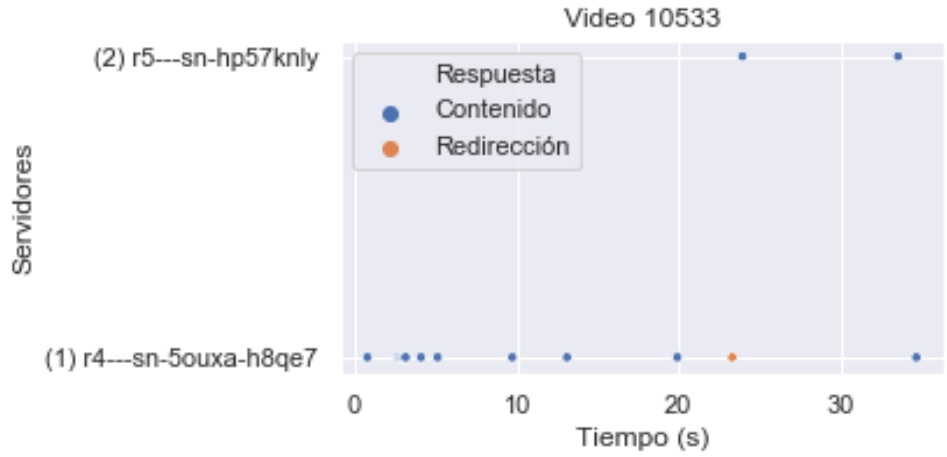

Figura 4.12: Gráfica de mensajes enviados a los servidores para el video 10533. En paréntesis se encuentra el nivel de jerarquía de los servidores.

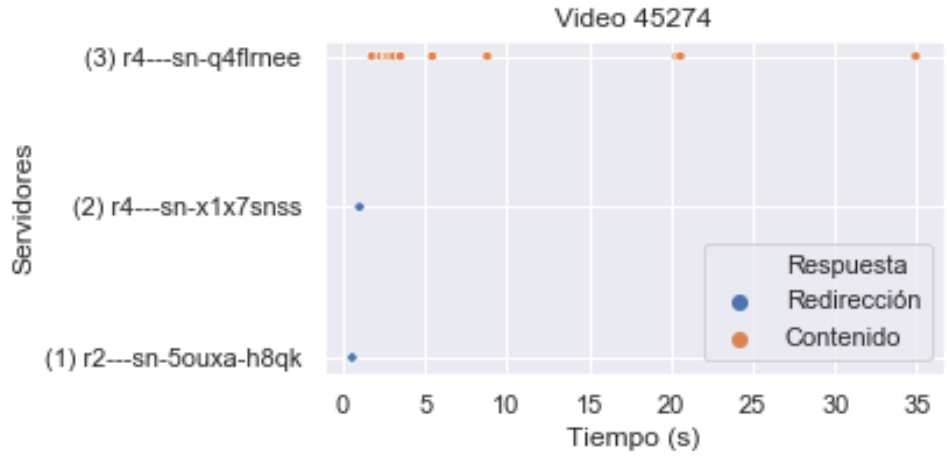

Figura 4.13: Gráfica de mensajes enviados a los servidores para el video 45274. En paréntesis se encuentra el nivel de jerarquía de los servidores.

que el segundo ocurre en la Figura 4.14. En la misma, se recibe un mensaje de redirección al segundo nivel de la jerarquía al comienzo de la reproducción. Desde el segundo nivel, primero se recibe contenido y luego una redirección hacia el tercer nivel. Finalmente, el resto del video es enviado desde el segundo y el tercer nivel de la jerarquía.

Finalmente, para la última pregunta, se busca entre las reproducciones en las que se contactó al servidor secundario por problemas de conectividad, si existe alguna en la que se vuelva a contactar al principal. El resultado es afirmativo, y un ejemplo es la Figura 4.15. Se puede ver que al comienzo el contenido se recibe desde un servidor en el segundo nivel de jerarquía, que resulta ser su servidor secundario, y luego se comienza a recibir contenido desde el servidor **r2—sn-ouxa-h8q6** que es el principal para ese video. Además, se puede ver que el tiempo que pasa hasta que comienza la reproducción del video es de casi 10 segundos, mostrando un

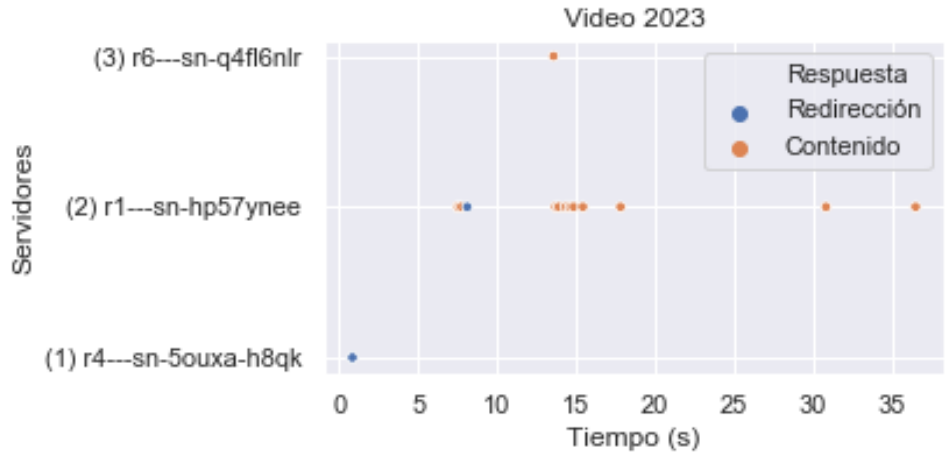

Figura 4.14: Gráfica de mensajes enviados a los servidores para el video 2023. En paréntesis se encuentra el nivel de jerarquía de los servidores.

problema de conectividad.

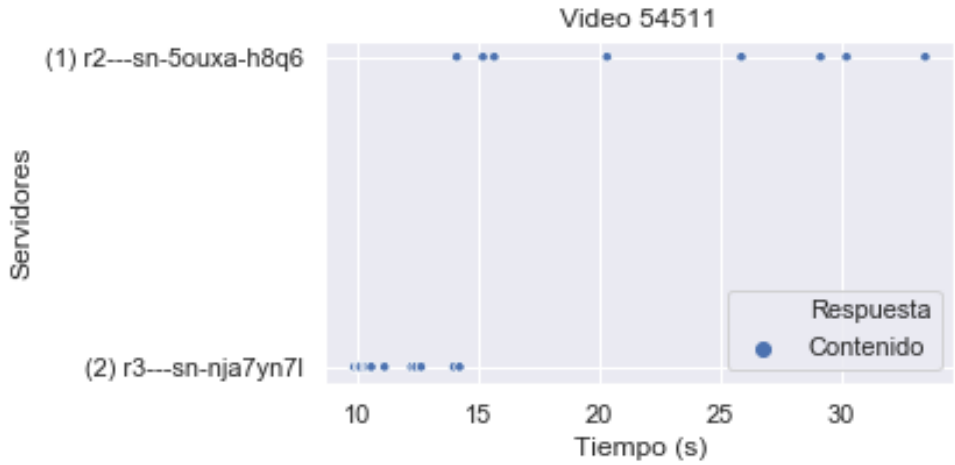

Figura 4.15: Gráfica de mensajes enviados a los servidores para el video 54511. En paréntesis se encuentra el nivel de jerarquía de los servidores.

4.4. Algoritmo de recomendación y cacheo

Se concluye esta sección con un breve análisis del algoritmo de recomendación y sistema de cacheo.

Los videos que se reprodujeron en el experimento eran los recomendados por YouTube. Sin un análisis muy profundo, se puede notar algo muy interesante del mismo: el algoritmo no se encuentra sesgado hacía contenido cercano geográficamente al usuario, o por lo menos no es su principal criterio de selección. Si esto fuera así, la cantidad de redirecciones posiblemente

hubiera sido mucho menor o directamente no hubiera ocurrido ninguna.

Utilizando el identificador del video de YouTube guardado para cada reproducción, es posible investigar si los videos que se reprodujeron más de una vez tuvieron redirecciones, y obtener información sobre el algoritmo de cacheo. Se toma el conjunto de datos y se filtra, obteniendo solamente los videos que se reprodujeron más de una vez, y, que en alguna de sus reproducciones haya ocurrido una redirección. Finalmente, se analizan los casos resultantes, donde un caso se refiere al conjunto de reproducciones para un video dado (existe un caso por identificador de video).

En el **48 %** de los casos, existió una redirección solamente la primera vez que se reprodujo el video. En el **22 %** siempre existió una redirección y el restante **30 %** representan casos en los que ocurrieron redirecciones, pero no se encuentran en los casos anteriores (un ejemplo de esta categoría puede ser un video con 10 reproducciones, que tuvo redirecciones en las reproducciones número 5 y 8). Esto muestra que el algoritmo de cacheo no utiliza el mismo time to live para todos los videos.

Capítulo 5

Factores que influyen en la redirección de un video

Se intentan descubrir factores que puedan incidir en la redirección un video, analizando los atributos obtenidos en el experimento. Esto también puede verse como una búsqueda de que atributos pueden indicar si un video está cacheado en Uruguay.

5.1. Popularidad

Antes de estudiar si la popularidad de un video es una causa que podría generar una redirección, es necesario establecer que es un video popular. Entre los atributos obtenidos del experimento, los que podrían definir la popularidad son: la cantidad de "me gusta", cantidad de "no me gusta", cantidad de visitas y cantidad de comentarios. Se va a realizar un estudio de cada uno de los atributos por separado.

Para facilitar el estudio, se van a utilizar gráficas de la función de distribución acumulada (CDF). La función de probabilidad acumulada para una variable aleatoria X esta dada por la fórmula:

$$F_X(x) \;=\; P(X \leq x)$$

Es decir, la gráfica de la función de distribución acumulada permite obtener la probabilidad de que una variable aleatoria tome valores menores o iguales a x.

Si para un atributo se realiza esta gráfica separando en las reproducciones que tuvieron redirecciones y las que no, se puede obtener información sobre cómo influye ese atributo en la ocurrencia de redirecciones. Si las curvas tienen una forma similar, quiere decir que ese atributo no influye en si el usuario es redirigido o no, mientras que, en el caso contrario, permite comparar los valores que toma el atributo cuando ocurren redirecciones y cuando no ocurren. La probabilidad va a ser estimada tomando casos favorables dividido el total de casos.

En la Figura 5.1 se encuentra la gráfica CDF de la cantidad de visitas de un video. Se aprecia que entre ambas curvas existe una diferencia bastante amplia, lo que indica que el atributo influye en la posibilidad de una redirección. Como la curva que representa a los videos con redirección (de color naranja) tiene un crecimiento más rápido al comienzo, se puede concluir que los videos con redirecciones contienen una cantidad de visitas menor. Por ejemplo, si se toma un video que tuvo redirecciones, la probabilidad estimada de que tenga 100000 o menos es de aproximadamente un 60 %, mientras que si no ocurrieron redirecciones esa probabilidad es de tan solo 40 %.

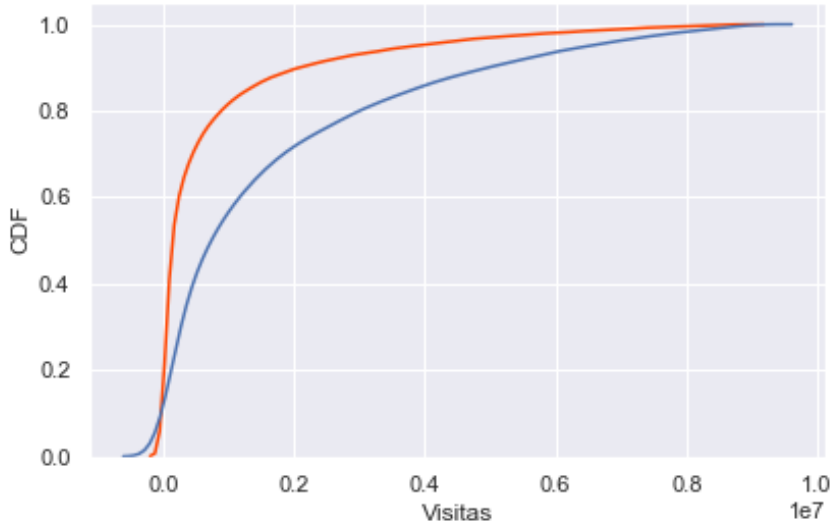

Figura 5.1: Gráfica CDF de la cantidad de visitas de los videos reproducidos. La curva de color naranja corresponde a los videos que tuvieron redirecciones. La azul se corresponde con los que no.

Repitiendo el mismo análisis pero para el atributo cantidad de "me gusta", cuya gráfica CDF se encuentra disponible en la Figura 5.2, se aprecia un comportamiento distinto. En este caso, las curvas son bastante similares, aunque nuevamente la curva de cuando ocurrieron redirecciones (en color naranja) se encuentra ligeramente por encima de la otra, mostrando una pequeña relación con la ocurrencia de redirecciones. Sin embargo, esta diferencia es muy pequeña como para poder concluir que es un atributo que influye en la posibilidad de una redirección, quedando solamente en una sospecha.

En el caso de la cantidad de "no me gusta", analizando la Figura 5.3 que muestra la gráfica de CDF de dicho atributo, se observa que es bastante parecida a la gráfica CDF de la cantidad de "me gusta". Por lo tanto, se concluye que existe la posibilidad de que la cantidad de "no me gusta" influya en la ocurrencia de redirecciones, pero sin evidencia suficiente para concluirlo.

Finalmente, con la cantidad de comentarios ocurre algo similar a los dos atributos anteriores,

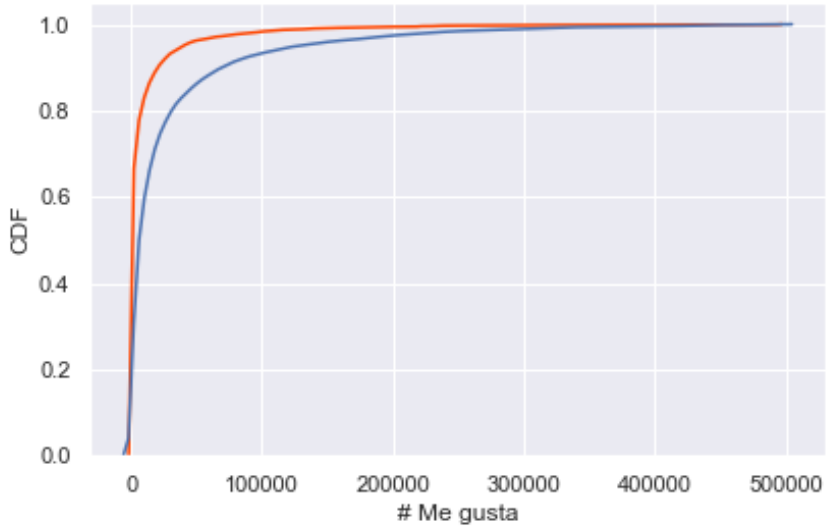

Figura 5.2: Gráfica CDF de la cantidad de "me gusta" de los videos reproducidos. La curva de color naranja corresponde a los videos que tuvieron redirecciones. La azul se corresponde con los que no.

como se puede ver en la Figura 5.4. Es decir, existe la posibilidad de que la cantidad de comentarios influya en la ocurrencia de redirecciones, pero no se puede asegurar.

La conclusión que se obtiene de esta sección es que, si bien todos los atributos relacionados a la popularidad presentan una pequeña relación con la ocurrencia de redirecciones, el único atributo que se puede asegurar que influye en la ocurrencia de las mismas es la cantidad de visitas.

5.2. Información sobre la red

El conjunto de datos contiene dos atributos que brindan información sobre la red: **stats_connection** y **time_to_first_byte** (tiempo hasta el primer byte) estos atributos se obtienen en el momento que el video se está reproduciendo, por lo que no son útiles para predecir una redirección, pero se investiga si pueden llegar a ser útiles para identificar si ocurrió una redirección. Para este análisis, se decidió dejar afuera los videos cuyo tiempo al primer byte fuera mayor a diez segundos, considerando un tiempo mayor a ese como un problema de conectividad del equipo.

Nuevamente se realizan gráficas de la función de distribución acumulada, esta vez del tiempo al primer byte (Figura 5.5) y stats_connection (Figura 5.7).

La gráfica de la distribución acumulada del tiempo al primer byte en la Figura 5.5, muestra que la curva asociada a los videos que no redireccionaron al usuario (en color azul) se encuentra por encima de la que corresponde a los videos que redireccionaron (naranja). Una

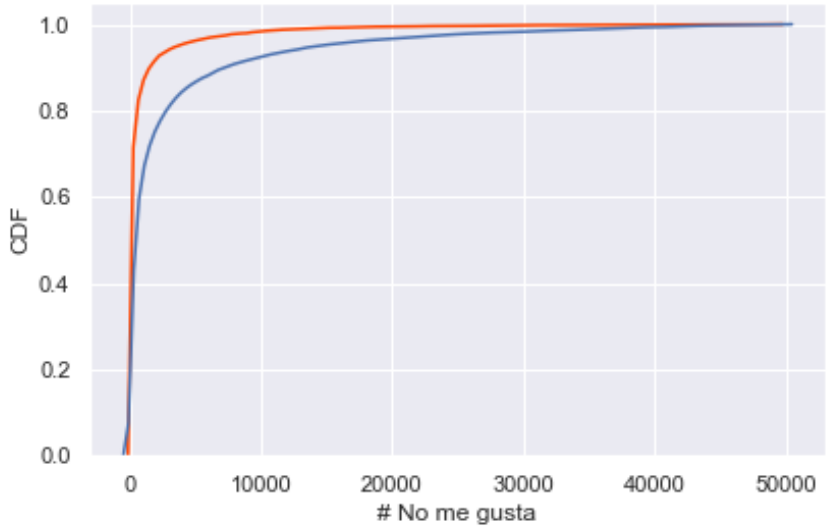

Figura 5.3: Gráfica CDF de la cantidad de "no me gusta" de los videos reproducidos. La curva de color naranja corresponde a los videos que tuvieron redirecciones. La azul se corresponde con los que no.

consecuencia de esto, es que el atributo tiempo al primer byte es útil para identificar la ocurrencia de redirecciones. Si se toma un video que no redirigió al usuario, la probabilidad de que su tiempo al primer byte haya sido menor a 2 segundos es de 60 %.

Además, se puede realizar una comparación con el artículo [4] previamente mencionado en la sección de antecedentes, dado que realiza un estudio similar. En el mismo, los autores calculan el *tiempo de inicialización*, definido como el tiempo que demora el equipo desde que realiza la primera solicitud a un servidor de contenido hasta que termina de descargar 1 MB de video. La gráfica CDF correspondiente a los resultados que obtuvieron en uno de sus equipos se puede observar en la Figura 5.6. La misma tiene un cierto parecido con la obtenida en este proyecto. Más aún, los autores llegan a las mismas conclusiones para ese atributo que las obtenidas en este trabajo para el atributo tiempo al primer byte.

Para el otro atributo en esta categoría, stats connection, la diferencia entre las curvas de su gráfica CDF cuando ocurren redirecciones (color naranja) y cuando no (color azul), visible en la Figura 5.7, es bastante grande. Esto quiere decir que es un atributo que brinda indicios sobre si ocurrió una redirección. Para el valor 20000 kbps, la curva que representa los videos con redirecciones (en color naranja) enseña que, dado un video con una redirección, hay una probabilidad de casi 80 % de que su atributo stats_connection sea menor o igual a 200000, mientras que si no ocurrieron redirecciones esa probabilidad es de tan solo 50 %, aproximadamente.

Sin embargo, es necesario tomar cierta precaución al tratar con este atributo, debido a la forma en la que se obtiene. La medida no se realiza en el mismo tiempo de comenzada la carga

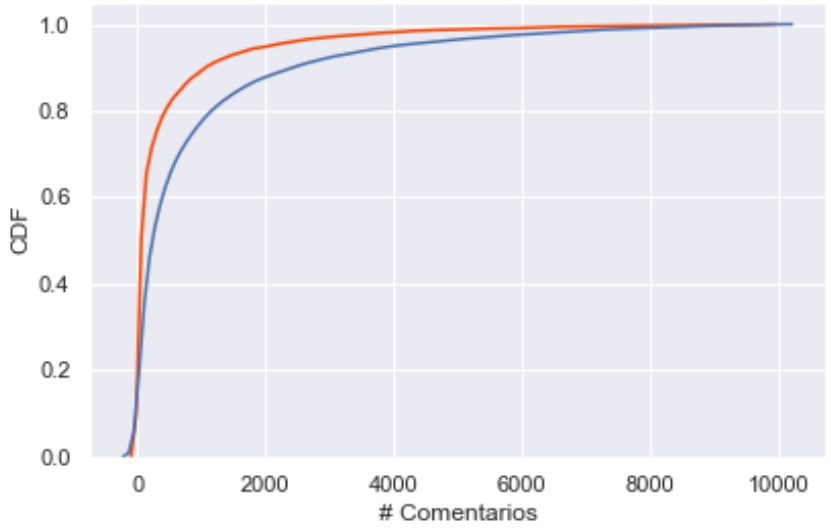

Figura 5.4: Gráfica CDF de la cantidad de comentarios de los videos reproducidos. La curva de color naranja corresponde a los videos que tuvieron redirecciones. La azul se corresponde con los que no.

del sitio para todos los casos. Además, no se sabe si las limitaciones de este atributo están dadas por la velocidad de conexión del usuario, la del servidor, u otros factores. Por ejemplo, la calidad del video, o puede suceder que YouTube mantenga la velocidad más baja posible, de forma de permitir al usuario mirar el video sin que se vacíe el buffer, lo que permitiría ahorrar ancho de banda, tanto a YouTube como al usuario.

5.3. Metadatos sobre los videos

Se analiza la posible influencia de otros atributos como la categoría, el idioma y la duración.

Comenzando con la duración, la Figura 5.8 muestra la gráfica CDF de la misma. Se puede observar que ambas curvas son idénticas, por lo que se puede concluir que este atributo no influye en la posibilidad de la redirección de un usuario.

Analizando las categorías, la cantidad de videos en cada una varía bastante, como se pudo observar anteriormente en el Cuadro 4.1. A efectos de realizar una comparación, se decide tomar una muestra de quinientos videos de cada categoría, utilizando solo las categorías que tengan por lo menos esa cantidad de videos. Los resultados arrojan que la probabilidad de redirección cambia según la categoría. Esto se puede ver en el Cuadro 5.1.

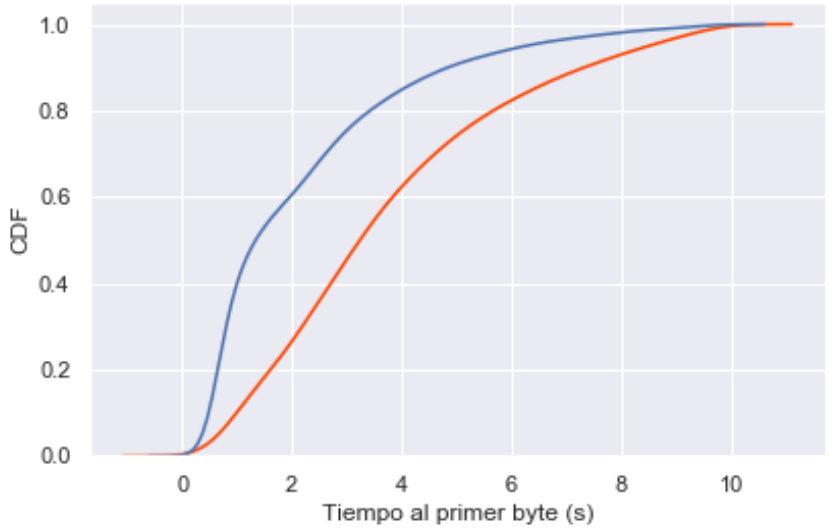

Figura 5.5: Gráfica CDF del tiempo al primer byte de los videos reproducidos. La curva de color naranja corresponde a los videos que tuvieron redirecciones. La azul se corresponde con los que no.

Categoría	**% Redirección**
Entretenimiento (24)	17.2 %
Personas y Blogs (22)	21 %
Música (10)	13.2 %
Juegos (20)	29.8 %
Películas y Animación (1)	22 %
Educación (27)	12.8 %
Deportes (17)	21.2 %
Noticias y Política (25)	37.8 %
Comedia (23)	30.6 %
Activismo (29)	41.2 %
Ciencia y Tecnología (28)	46.8 %
Vehículos (2)	37.4 %
Mascotas y Animales (15)	37.2 %

Cuadro 5.1: Porcentaje de redirección en cada categoría, tomando una muestra de quinientos videos en cada categoría. El número en paréntesis es el identificador de la categoría utilizado por la API de YouTube.

Las categorías de música y educación son las que tienen una menor cantidad de videos redireccionados, con una probabilidad promedio de 13 %, mientras que las de noticias y política, activismo, vehículos, ciencia y tecnología, y, mascotas y animales, son las que tuvieron una mayor probabilidad de redirección, con probabilidades que van desde 37.4 % hasta 46.8 %. Cabe destacar que dos de las categorías con más probabilidad de redirección contienen videos que se encontraban en las semillas iniciales. Se puede decir que la categoría de un video

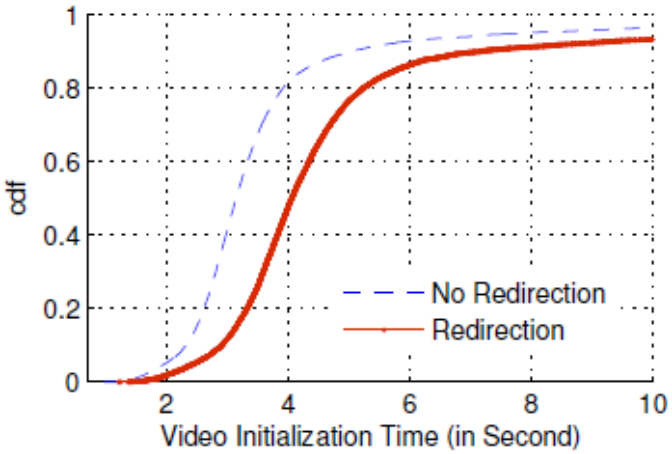

Figura 5.6: Gráfica CDF del tiempo de inicialización. Por V. K. Adhikari, S. Jain, Y. Chen, and Z. L. Zhang, *Vivisecting YouTube: An active measurement study*, 2012, Proceedings IEEE INFOCOM.

influye en la posibilidad de una redirección.

Considerando la definición (del video), los videos en alta definición tienen un 20 % de probabilidad de redirección y los videos en baja definición 26 %. Estos números son bastante similares. Si el usuario es redirigido o no, no parece verse influenciado por la definición del video. Sin embargo, para algún trabajo futuro se podría investigar si cuando YouTube cachea un video, lo hace para todas las definiciones disponibles.

Finalmente, el idioma parece influir en la ocurrencia de redirecciones. Los porcentajes en el Cuadro 5.2 no son sorprendentes, es de esperar que el idioma menos redirigido sea el español, dado que es el que se habla en Uruguay. Para el alemán, portugués y el conjunto de otros idiomas, la probabilidad es bastante similar, de alrededor de 27 %. Se destaca que la probabilidad de redirección de los videos en holandés sea bastante más alta que la de los videos en alemán, y los idiomas en la categoría otros, siendo que se esperaría que todos esos idiomas tuvieran la misma popularidad en Uruguay.

Idioma	% Redirección
Holandés	44 %
Otros	28 %
Portugués	27 %
Alemán	26 %
Inglés	18 %
Español	12 %

Cuadro 5.2: Porcentaje de redirección según el idioma del video.

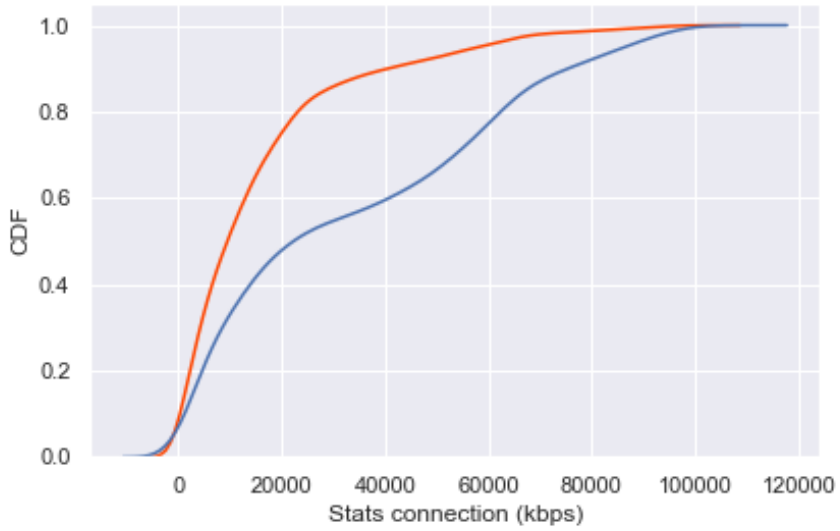

Figura 5.7: Gráfica CDF del atributo stats_connection. La curva de color naranja corresponde a los videos que tuvieron redirecciones. La azul se corresponde con los que no.

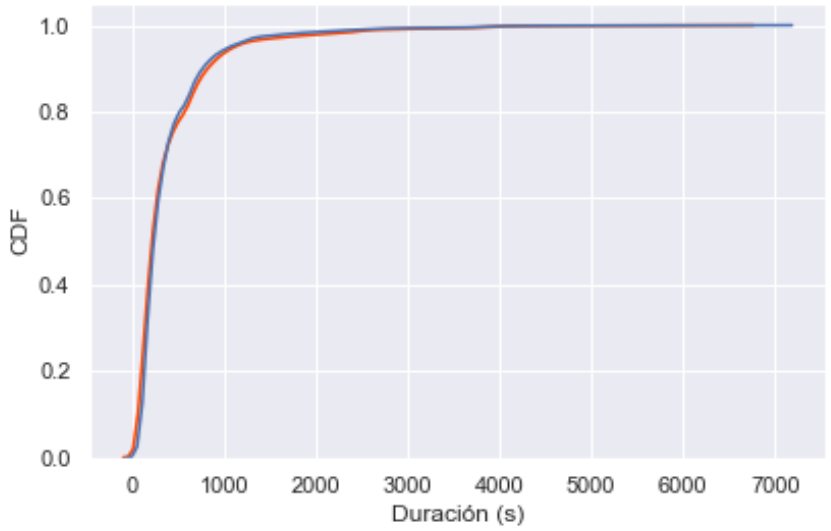

Figura 5.8: Gráfica CDF de la duración de los videos reproducidos. La curva de color naranja corresponde a los videos que tuvieron redirecciones. La azul se corresponde con los que no.

Capítulo 6

Conclusiones

En este proyecto se logró investigar ciertas partes del funcionamiento de YouTube. Se pudo generar un conjunto de datos obtenidos al reproducir videos de YouTube, junto con los servidores de los cuales provino cada video, que luego fue analizado. Se considera que este conjunto de datos es capaz de brindar aún más información que la obtenida a lo largo del presente proyecto. En particular, se puede intentar investigar las peticiones a los servidores de contenido que no contienen datos, mencionadas en la sección de resultados, que parecen ser una forma de informar a YouTube la calidad del servicio, debido a que el recurso que se solicita es */videogoodput*.

Se logró obtener valiosa información sobre la procedencia del contenido, se comprobó la utilización de cachés en Uruguay y la presencia de una jerarquía de servidores, que ya había sido descubierta en investigaciones previas, junto con la ubicación geográfica de todos los servidores. Los distintos niveles de la jerarquía son utilizados cuando el video no se encuentra en el servidor a consultar y el usuario es redirigido a un nuevo servidor. Nuevamente, este mecanismo ya había sido mencionado en investigaciones previas, y se comprobó que aún se utiliza.

Se destaca que los servidores de jerarquía más alta, se encontraron todos en Uruguay, es decir, al mirar un video, el primer servidor consultado siempre está en Uruguay. Cuando el video no se encuentra, el usuario puede ser redirigido a servidores en Brasil, Chile, Canadá, Francia, Italia, Estados Unidos, Los Países Bajos o Argentina.

Buscando profundizar un poco más en el estudio de la jerarquía, se construyó un grafo que permite ver que servidor redirige a cuál. El análisis de este grafo sirvió de ayuda para descubrir servidores con mayor importancia a nivel de conectividad.

Alguna de las conclusiones que este estudio permitió obtener es que el idioma de un video, que se asignó como el mismo idioma que el de su título, no parece tener una relación directa con el idioma hablado en el país que se encuentra el servidor del que provino.

Se implementó un método de geolocalización que utiliza el retardo de propagación entre un

objetivo a encontrar y varios equipos. En este caso, los equipos utilizados fueron sondas pertenecientes a la red de RIPE Atlas. Este método no utiliza medidas entre los equipos utilizados para geolocalizar, lo cual le permite utilizar menos mediciones y brinda la posibilidad de cambiar de probes en cada medida de geolocalización de manera veloz.

Finalmente, se intentó investigar que atributos de los obtenidos son capaces de influir en la posibilidad de que el usuario sea redireccionado a otro servidor, o dicho de otra manera si el video va a estar cacheado en Uruguay o no. Se descubre que, cuanto mayor sea la cantidad de visitas del video, más probable es que provenga de nuestro país. Además, la categoría del video y el idioma también parecen afectar en este sentido. Sin embargo, también se encuentran atributos que no influyen, como la duración del video.

En comparación con los artículos analizados en los antecedentes, se puede decir que, si bien cambiaron los dominios utilizados por YouTube, el mecanismo de redirecciones sigue funcionando de manera similar, por lo menos a nivel de capa de aplicación. No se tiene información suficiente para concluir si los servidores DNS juegan un rol importante en este tema, como era mencionado por los artículos.

6.1. Trabajo Futuro

Como se mencionó anteriormente, se cree que el conjunto de datos recopilado todavía es capaz de brindar más información acerca de YouTube. Se pueden investigar las peticiones a los servidores de contenido que no contienen datos multimedia o tratar de explicar cuándo es que una redirección a nivel de capa de aplicación ocurre mediante un código de estado de HTTP de redirección o mediante un nuevo servidor en el cuerpo de la respuesta, por ejemplo.

Además, de los atributos obtenidos para cada reproducción, hubo algunos que no fueron utilizados. Uno de ellos fue la hora en la que se reprodujo el video. Esto podría brindar información sobre horas en las que ocurren más redirecciones y tratar de explicar si coincide con algún evento del día como puede ser la hora de salida de las escuelas, por poner un ejemplo.

Otro atributo interesante de analizar, es el próximo video a reproducir. Si bien dado un video no se tiene la clave primaria del siguiente video que se reprodujo ni el anterior, utilizando el atributo del próximo video a reproducir se puede intentar generar la cadena de videos reproducidos. Esta información puede resultar útil para intentar estudiar y comprender el algoritmo de recomendación de videos.

Finalmente, puede resultar útil repetir el experimento desde otros lados del mundo. De esta manera, se puede comparar con los resultados obtenidos para Uruguay, e intentar reproducir videos relevantes a Uruguay (por ejemplo, utilizando las semillas iniciales en esa categoría) y ver si en algún momento llega contenido desde nuestro país. Algunos lugares interesantes para probar pueden ser Estados Unidos, o algún país ubicado en Asia. El lugar primer lugar resulta de observar la gran cantidad de servidores ubicados en ese país, una interrogante

que se puede plantear es investigar si aun así viene contenido de otro país. El segundo lugar surge de no haber encontrado ningún servidor en ese continente, a pesar de localizarse dos datacenters de Google en el mismo, uno en Taiwán y otro en Singapur.

Bibliografía

[1] *Cisco Visual Networking Index: Forecast and Methodology, 2016–2021*. en. URL: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html (visitado 07-11-2018).

[2] James F. Kurose y Keith W. Ross. *Computer Networking: A Top-Down Approach (6th Edition)*. 6th. Pearson, 2012. ISBN: 978-0-13-285620-1.

[3] R. Torres y col. "Dissecting Video Server Selection Strategies in the YouTube CDN". En: *2011 31st International Conference on Distributed Computing Systems*. Jun. de 2011, págs. 248-257. DOI: 10.1109/ICDCS.2011.43.

[4] V. K. Adhikari y col. "Vivisecting YouTube: An active measurement study". En: *2012 Proceedings IEEE INFOCOM*. Mar. de 2012, págs. 2521-2525. DOI: 10.1109/INFCOM.2012.6195644.

[5] P. Hillmann y col. "Modelling of IP geolocation by use of latency measurements". En: *2015 11th International Conference on Network and Service Management (CNSM)*. Nov. de 2015, págs. 173-177. DOI: 10.1109/CNSM.2015.7367355.

[6] B. Gueye y col. "Constraint-Based Geolocation of Internet Hosts". En: *IEEE/ACM Transactions on Networking* 14.6 (dic. de 2006), págs. 1219-1232. ISSN: 1063-6692. DOI: 10.1109/TNET.2006.886332.

[7] P. Enge y P. Misra. "Special Issue on Global Positioning System". En: *Proceedings of the IEEE* 87.1 (ene. de 1999), págs. 3-15. ISSN: 0018-9219. DOI: 10.1109/JPROC.1999.736338.

[8] Venkata N. Padmanabhan y Lakshminarayanan Subramanian. "An Investigation of Geographic Mapping Techniques for Internet Hosts". En: *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. SIGCOMM '01. New York, NY, USA: ACM, 2001, págs. 173-185. ISBN: 978-1-58113-411-7. DOI: 10.1145/383059.383073. URL: http://doi.acm.org/10.1145/383059.383073 (visitado 06-11-2018).

[9] *RIPE Network Coordination Centre*. URL: https://www.ripe.net (visitado 08-11-2018).

[10] *RIPE Atlas*. URL: https://www.ripe.net/analyse/internet-measurements/atlas (visitado 08-11-2018).

[11] *Google Edge Network*. URL: https://peering.google.com/#/infrastructure (visitado 07-11-2018).

[12] *PeeringDB*. URL: https://www.peeringdb.com/net/1495 (visitado 07-11-2018).

[13] *HTTP Archive (HAR) format*. URL: https://dvcs.w3.org/hg/webperf/raw-file/tip/specs/HAR/Overview.html (visitado 10-11-2018).

[14] J. Chen y col. "A Landmark Calibration Based IP Geolocation Approach". En: *2015 10th International Conference on Availability, Reliability and Security*. Ago. de 2015, págs. 411-416. DOI: 10.1109/ARES.2015.52.

[15] *seaborn: statistical data visualization — seaborn 0.9.0 documentation*. URL: https://seaborn.pydata.org/ (visitado 16-11-2018).

Apéndice A

Implementación algoritmo de geolocalización

Se presenta un pseudo-código extendido del algoritmo de geolocalización.

```
geolocalizar(objetivo) {
   probes = [ ];

   for ubicacion in ubicaciones {
      probes.append(API_RIPE.get_probes(ubicacion));
   }

   medida = API_RIPE.crear_medida(probes, objetivo, tipo='ping');
   // medida contiene el identificador unico de la medida.

   // Esperar a que se complete la medida
   pings = API_RIPE.get_results(medida);

   min_ping = MAX_INT
   min_probe = 0

   for ping in pings {
      id = ping.probe_id
      // min contiene el valor minimo obtenido para el ping

      if ping.min == -1 {
      // -1 indica error
         continue
      }

      if ping.min < min_ping {
      // se encontro un ping menor al encontrado hasta el momento
         min_ping = ping.min
         min_probe = ping.probe_id
      }
   }
```

```
    // min_ping contiene el ping mas chico encontrado
    // min_probe el id de la probe con el menor ping

    // se obtiene el objeto probe, que contiene sus coordenadas
    min_probe = API_RIPE.get_probe(min_probe)

    // velocidad de la luz (km/s)
    C = 300000

    latitud = min_probe.geometry.coordinates.latitude
    longitud = min_probe.geometry.coordinates.longitude

    // se convierte el retardo de propagacion en distancia
    // se multiplica por 0.001 por las unidades
    // min_ping esta en ms
    distancia = (min_ping/2) * (4/9) * C * 0.001

    // Se guarda en la base de datos: objetivo, medida, latitud, longitud,
       distancia, min_ping y probe

}
```

Apéndice B

Implementación del cliente

En este apéndice se tratará explicar con más detalles la implementación del programa cliente, el encargado de navegar por el sitio de YouTube consumiendo videos, recolectando datos y enviándolos a un servidor.

Durante el desarrollo se encontraron diversos problemas que dificultaron la implementación. Fue necesario cambiar las configuraciones por defecto de Selenium y BrowserMob para solucionar los inconvenientes.

La configuración por defecto del proxy no decodifica respuestas enviadas en formato Brotli, un algoritmo de compresión de tráfico creado por Google. Las respuestas que se detectaron en este formado y resultaban de interés en poder comprender eran las respuestas a las peticiones **/watch**, en las que se enviaba datos para comenzar a reproducir un video nuevo.

Como consecuencia, fue necesario procesar la respuesta para decodificarlo. Además, antes de enviar los datos al servidor, el archivo HAR se procesaba y se eliminaba contenido innecesario, como imágenes.

La conexión con el servidor implicó la necesidad de utilizar un túnel SSH para los equipos que se encontraban fuera de la facultad. Para los equipos encontrados en la facultad, fue necesario utilizar además de BrowserMob, el proxy de la facultad.

Además, fue necesario capturar excepciones y manejarlas adecuadamente para que el cliente continuara funcionando luego de cualquier problema.

El navegador utilizado fue Firefox, debido a que era el más sencillo de configurar.

```
navegar(seed, ubicacionPC) {
   proxy = new BrowserMobProxy();
   if (ubicacionPC == 'fing') {
      proxy.setProxy('proxy.fing.edu.uy', puerto=3128)
```

```
}

proxy.start();
proxy.capturarHar();

Navegador = new FirefoxDriver(proxy=proxy);

proxy.newHar();
primer_video = true;
navegador.get(seed);

while(true) {
   // esperar a que cargue el sitio

   video = new Video();
   // se saca el ID del video utlizando la url
   // y expresiones regulares
   video.id = obtener_id(navegador.url);


   if (primer_video) {
      // si es el primer video desde que se cargo la pagina
      // se abren las estadisticas
      // implica ejecutar un click derecho sobre el reproductor
      // seleccionar el ultimo elemento dentro el menu
      // y realizar un click
      abrir_estadisticas();
      primer_video = false;

      // el panel no se cierra al cambiar de video
   }

   panel_estadisticas =
       navegador.cssSelector('.html5-video-info-panel-content').toArray();

   // se procesa los elementos del arreglo panel_estadisticas, // extrayendo
       las estadisticas que se deciden almacenar.
   // por ejemplo stats_quality se obtiene de la siguiente manera
   video.stats_quality = panel_estadisticas[2].findByTag("span")).getText();

   // Llamada a la API de YouTube para obtener el resto de los datos
   // me gusta, no me gusta, cantidad de comentarios, etc.

   datosVideo = API_YOUTUBE.getVideo(id);
   // se agregan los valores necesarios de datosVideo a video
```

```
      // se itera sobre las paginas del archivo HAR eliminando datos innecesarios
      HAR = proxy.getHar();
      for(entrada in HAR) {
         mimeType = entrada.mimeType;
         content_encoding = entrada.content_encoding;

         if(mimeType == 'audio' || mimeType == 'image' || mimeType = 'video') {
            entrada.text = '';
         }

         if(content_encoding == 'br') {
            entrada.text = decodificar_brotli(entrada.text);
         }
      }
      video.har = HAR;

      if(ubicacionPC != 'fing') {
         // crear tunel ssh
      }

      enviar_datos_al_servidor(video);

      // buscar el primer video recomendado usando selectores css
      proxy.new_har();
      proximo_video.click();
   }
}
```

Se omitió exponer el manejo de excepciones para que haya una mejorar compresión del código.

Apéndice C

Implementación del servidor

Similar a lo realizado con el programa cliente, se expone un pseudo-código del servidor.

```
@route('/')
agregar_servidor(peticion){
   // los datos se enviaban en formato JSON
   datos = json.loads(peticion.data)
   procesar_datos(datos);
}

procesar_datos(video){
   // la variable video contiene un diccionario donde todos los valores, excepto
      el archivo HAR pertenecen a la base de datos, y por lo tanto, el unico
      procesamiento que es necesario hacer es sacar la captura HAR.

   har = video.pop('har');
   // es necesario obtener los servidores a partir de la captura
   servidores = {};
   for(entrada in har) {
      // se utilizan expresiones regulares para capturar solamente las entradas
         con peticiones a direcciones .googlevideo.com
      if(es_servidor_contenido(entrada.url)) {
         dominio = procesar_url(entrada.url);
         // se mira si es /videoplayback o /videogoodput
         tipo = get_tipo(entrada.url);

         // se guarda en el diccionario el tipo de servidor y la cantidad de bytes
            enviados. En caso de ya existir ese servidor, se sobreescribe el
            tipo, pero la cantidad de bytes se suma.
         servidores[dominio]['tipo'] = tipo;
         servidores[dominio]['cantidad_bytes'] += entrada.response.bodySize;
      }
   }
```

```
   // video y servidores se guardan en la base de datos. har se guarda en un
       archivo de texto.
   almacenar_video(video,servidores,har);

   // geolocaliza en un thread nuevo
   start_thread(proceso_geolocalizacion, servidores.keys());
}

proceso_geolocalizacion(servidores) {
   for(servidor in servidores){
      if (!esta_localizado(servidor)) {
         geolocalizar(servidor);
      }
}
```

Se comentan algunas aclaraciones con respecto al servidor. No todos los atributos fueron obtenidos en el momento en que se recibe la petición del cliente. Esto sucedió debido a que hubo atributos, como el tiempo al primer byte, que se decidieron extraer luego de finalizado el experimento. El procedimiento para obtener esos atributos es similar a la función procesar_datos. En vez de recibir una petición, los datos de la captura HAR se leen del disco del equipo, y se modifica el código del cuerpo del for, de manera de buscar lo deseado. Finalmente, en lugar de guardar el video, se actualiza el ya existente, agregando el nuevo atributo.

La función de geolocalización es similar a la que se encuentra en el apéndice A. La diferencia radica en que existe un proceso ejecutándose en segundo plano que obtiene probes y las almacena en un archivo cada 45 minutos. En lugar de pedir probes nuevas para cada objetivo, la función geolocalizar las obtiene de ese archivo. Esto ahorra peticiones a la API de RIPE y permite iniciar la medida de geolocalización con mayor rapidez, dado que al tener una gran cantidad de lugares a pedir probes, el proceso de obtener las mismas no es muy rápido.