Tesis de Grado de Ingeniería Electrónica

Tomografía de Internet: Adquisición y Estudio de Parámetros Dinámicos de la Red

Autor:
Anderson Ricci, Mauricio

Director de Tesis:
Dr. Ing. J.I. Alvarez-Hamelin

Año 2016

Índice general

Capítulo 1

Introducción

El objetivo del presente trabajo está puesto en la adquisición y análisis de datos de la topología de Internet, tanto a nivel IP como a nivel de *router*, mediante mediciones externas, temática conocida en la bibliografía como "Tomografía de Internet". Para esto hemos desarrollado un programa, al cual hemos denominado `Magallanes` [1], que permite realizar mediciones tomando como base la plataforma PlanetLab.

El presente trabajo continuará con tres capítulos en donde se introducirá el marco teórico del mismo. Luego se explicará el desarrollo de `Magallanes`, y por ultimo de dedicará un capitulo para la realización de una exploración completa, describiendo como se ha llevado a cabo y analizando los resultados obtenidos. Finalmente se cerrará el trabajo con un capitulo de conclusión y futuras lineas de trabajo.

En el primer capitulo posterior a la introducción hablaremos sobre los conceptos que se utilizarán. Una vez finalizada esta descripción comentaremos sobre los proyectos de investigación actuales y la importancia de estos para la comunidad científica y la industria de las telecomunicaciones.

Lo siguiente que haremos es dedicar un capitulo a los fundamentos de `Magallanes`. Hablaremos sobre los motivos que nos han llevado a su diseño, el entorno de trabajo sobre el que fue diseñado, la filosofía que seguimos al momento del diseño y la arquitectura resultante del mismo. También mencionaremos los problemas con que nos encontramos y las soluciones por las que hemos optamos.

Una vez descrito el funcionamiento de `Magallanes`, y el marco teórico de sus componente, llega el momento de ponerlo a prueba. Para lo mismo hemos dividido el capitulo en dos partes. La primera consiste en explicar la exploración más extensa que hemos realizado, detallando la preparación del entorno de trabajo, la configuración elegida y el análisis de los resultados. Luego de esto, hablaremos de como hemos utilizado `Magallanes` para la adquisición de datos utilizados en el desarrollo de un articulo sobre la influencia de tecnología MPLS en la topología de Internet [2].

Finalmente, cerraremos el presente trabajo con las conclusiones que hemos sacado del diseño de `Magallanes` y los resultados que nos ha entregado. A su vez, mencionaremos futuras lineas de trabajo que se pueden seguir a partir del mismo.

Capítulo 2

Estado del arte

En este capitulo nos encargaremos de dar una explicación a la razón que nos ha llevado al diseño de `Magallanes`, hablando de la estructura de Internet e introduciendo el concepto de "Tomografía de Internet". Al final del capitulo comentaremos sobre los proyectos de investigación actuales, y la importancia de estos para la comunidad científica y la industria.

2.1. Breve introducción a Internet

La presente tesis trata acerca del diseño de un programa para la adquisición de datos de topología de Internet. Por este motivo debemos comenzar hablando un poco sobre Internet.

Origen e impacto social

Indudablemente Internet es la red más importante que existe en la actualidad, aunque no es del todo una red, sino un inmenso conjunto descentralizado de redes de comunicación interconectadas entre sí que utilizan la familia de protocolos TCP/IP, lo cual garantiza que las redes físicas heterogéneas que la componen funcionen como una red lógica única de alcance mundial [3]. Estas redes de comunicación a las que hacemos referencia están administradas independientemente unas de otras, y se las conoce como *Sistemas Autónomos*.

Los orígenes de Internet se remontan a la década de 1960 cuando el departamento de defensa de EEUU buscaba una alternativa al sistema de comunicación basado en *conmutación de circuitos* imperante hasta el momento en vista de una posible guerra con la Unión Soviética. En ese contexto surgieron los primeros trabajos sobre el sistema de comunicación basado en *conmutación de paquetes*, el cual presentaba una clara ventaja en términos en robustez y tolerancia a fallas. En 1969 dentro del proyecto `Arpanet` surgió la primer red, la cual conectaba inicialmente nodos en el MIT y la UCLA, y que posteriormente en la década de 1980 llegó a formar parte de un *backbone* que conectaba diversas redes militares y académicas. En esa misma década se desarrolló el stack de protocolos TCP/IP que posibilitó la interconexión de

redes en todo el mundo gracias a la estandarización de los protocolos que se requerían para acceder a ella. Posteriormente, en los inicios de la década de 1990, comenzaron a aparecer los primeros ISP que proveían acceso a la red a mayor cantidad de grupos sociales. En este mismo tiempo surgió la *World Wide Web* junto con otro gran número de servicios que provocó que para fines de la década ya se hiciese notar un impacto tremendo en la cultura y el comercio [4].

Hoy en día es innegable el impacto que ha tenido Internet en la sociedad actual. Este impacto abarca casi todos los ámbitos humanos, desde la economía hasta el modo en que interaccionan las personas. En la figura 2.1 se observa el nivel de penetración en la sociedad en la actualidad. Respecto de la evolución que llevó a esto, de 2000 a 2009, el número de usuarios de Internet a nivel mundial aumentó 394 millones a 1858 millones, y posteriormente, desde 2010 a 2015 este número aumento un 753 %, implicando un nivel de penetración en la sociedad de cerca del 46.4 % [5]. Esto que se comenta es la razón por la cual existe una amplia comunidad científica interesada en su estudio en las diferentes temáticas que abarca.

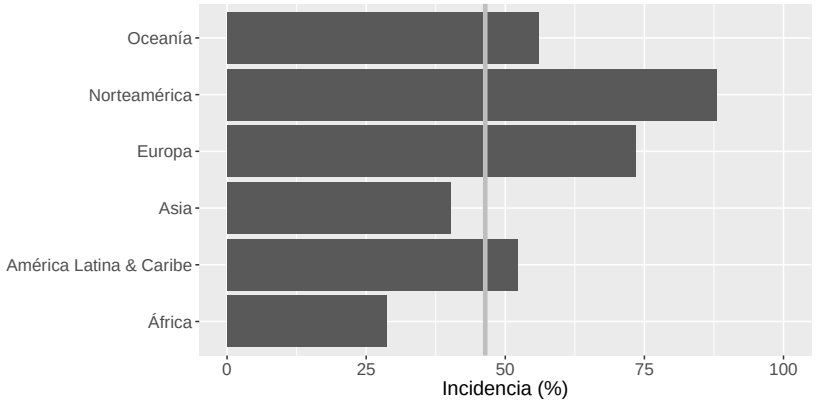


Figura 2.1: Penetración de Internet en la población mundial.
Datos por región - Noviembre 2015 [6].

Funcionamiento

Las comunicaciones a través de Internet son posibles gracias a la interacción de varias capas de abstracción, cada cual encargada de ciertas tareas e independiente en funcionamiento unas de otras, en las que hay presentes gran cantidad de equipos de comunicaciones y software especifico. Este conjunto de capas surgen del protocolo TCP/IP y se clasifican en:

- Capa de aplicación: ofrece a las aplicaciones la posibilidad de acceder a los servicios de las capas inferiores posibilitando comunicación de aplicaciones en entre dispositivos remotos.
- Capa de transporte: encargada de efectuar el transporte de datos entre dos dispositivos

independizandolo del tipo de red física que está utilizando. Los protocolos imperantes son TCP y UDP, el primero orientado a conexión y el otro sin conexión.

- Capa de red: se encarga de detectar los caminos existente entre redes y dirigir por estos los paquetes de datos. El objetivo de la capa de red es hacer que los datos lleguen desde el origen al destino, aún cuando ambos no estén conectados directamente
- Capa de acceso al medio: esta capa se ocupa del direccionamiento físico, del acceso al medio, de la detección de errores, de la distribución ordenada de tramas y del control del flujo.

Los temas tratados en esta tesis se desarrollan principalmente en la capa de red, por lo que nos enfocaremos en la explicación de esta. Todo surge de lo que mencionamos previamente, la clasificación según como se establece la conexión entre los dispositivos finales. Originalmente los primeros sistemas de comunicación, por ejemplo la red telefónica, para comunicar dispositivos finales establecían un circuito físico entre ellos que permanecía activo durante toda la comunicación haciendo uso exclusivo del medio. A este tipo de comunicaciones se las denomina *por conmutación de circuitos*. A pesar que presenta ciertas ventajas como la transmisión de datos en tiempo real y la simplicidad en la gestión de los nodos, posee ciertas desventajas que no la hacían apta para usos militares, principalmente en lo que se refiere a tolerancia a fallos y uso de recursos. Durante una comunicación se hace uso del medio físico en forma exclusiva y si surge una falla inmediatamente se corta la comunicación por lo se debe volver a establecer la misma desde el principio siempre y cuando allá disponibilidad en el medio. En la figura 2.2 se muestra un esquema de este tiempo de conexión.

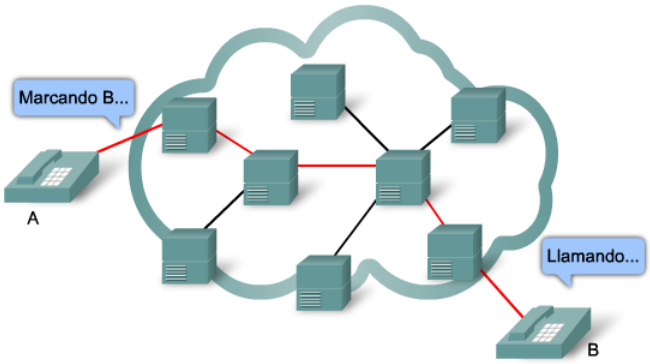


Figura 2.2: Ejemplo de conmutación por circuitos

Para solventar las limitación que este modelo presentaba se desarrolló la *conmutación por paquetes*. En este sistema el emisor divide los mensajes en un número arbitrario de paquetes que contienen los datos, donde adjunta a cada paquete una cabecera con la dirección origen

y destino así como datos de control que son utilizados por el medio para transmitir dichos paquetes. Este modelo presenta como ventaja que si hay errores en la comunicación, los datos perdidos se retransmiten automáticamente sin perdida de la conexión. Además se optimiza la utilización del medio de transmisión, que pasa a ser compartido, y en caso de perder alguno de los nodos durante una transmisión esta no se interrumpe sino que se redirige el trayecto que siguen los paquetes. En la figura 2.3 se muestra un esquema de este tiempo de conexión.

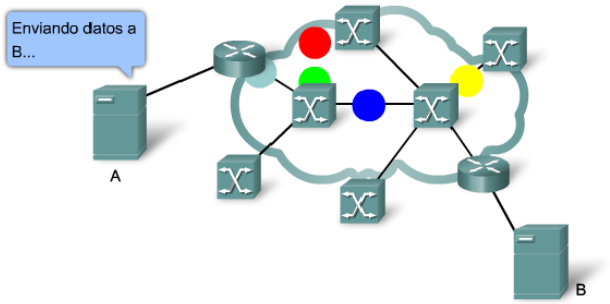


Figura 2.3: Ejemplo de conmutación por paquetes

Las redes de computadoras actuales se basan en la transmisión de datos mediante sistema de conmutación por paquetes, donde el protocolo TCP/IP es el encargado de indicar como deben ser generado y tratados dichos paquetes. Entonces, ya hemos comentado que Internet es una red de redes, pero aún no hemos explicado lo que es propiamente una red. Una *red de computadoras* es un conjunto de equipos informáticos y software conectados entre sí por medio de dispositivos físicos que envían y reciben impulsos eléctricos, ondas electromagnéticas o cualquier otro medio para el transporte de datos, con la finalidad de compartir información, recursos y ofrecer servicios [7]. Los componentes físicos principales que conforman una red son los *dispositivos finales*, los *switch* que conectan dispositivos dentro de una misma red a nivel de capa de enlace, y los *routers* que interconectan redes.

Los router son elementos que operan en la capa de Internet y se encargan de dirigir los paquetes desde un origen hasta un destino entre las diversas redes que hay entre ambos. La decisión de por cual ruta dentro de las disponibles utilizar para enviar los paquetes es tomada por el protocolo de enrutamiento configurado en cada uno. Para ejemplificar la idea de rutas hemos generado un `traceroute` desde un nodo de PlanetLab ubicado en Illinois (USA) hasta otro en Nueva Zelanda. Posteriormente, con los resultados del traceroute y la ayuda de una herramienta de geolocalización de IPs* y diversas bibliotecas de R† hemos generado la figura

*MaxMind. https://www.maxmind.com/

†Los paquetes de **R** utilizados han sido: `ggplot2`; `ggmap`; `geosphere`

2.4 donde se observa el camino seguido por el paquete generado en USA a través del planeta hasta su destino. En cada uno de los punto donde se observa un cambio de dirección lo que se encuentra es un router que tomó la decisión de enviar el paquete por uno u otro camino dentro de los disponibles.

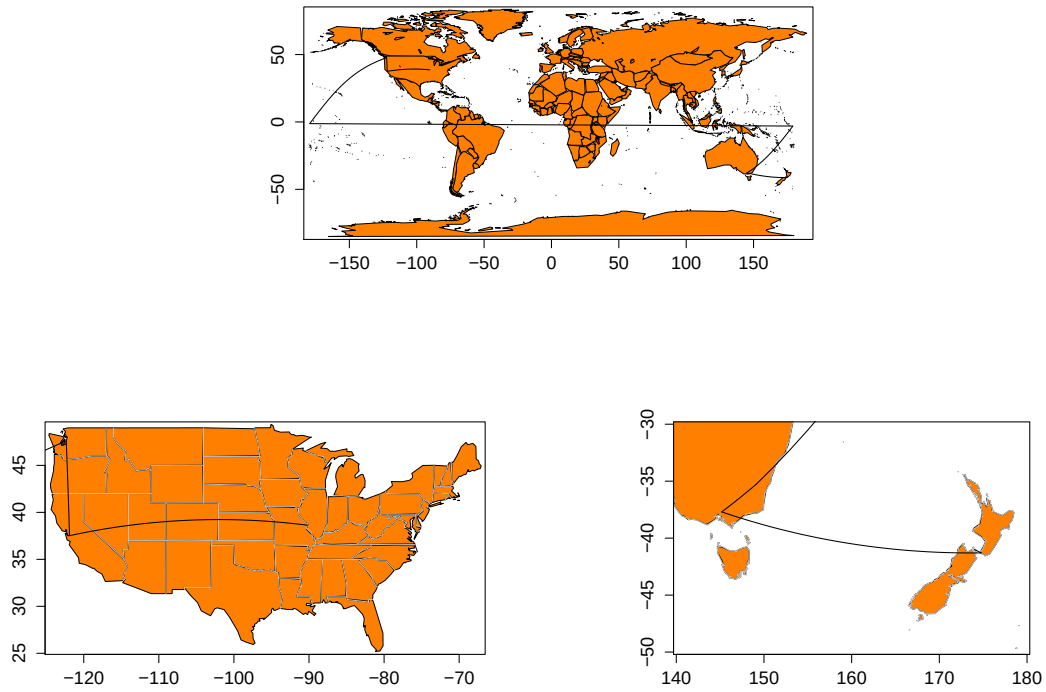


Figura 2.4: Ruta seguida por un paquete desde un nodo en EEUU hasta Nueza Zelanda

La importancia de estudiar las características de estos caminos se debe a que sobre Internet funcionan un gran número de servicio con alcance global que son los hacen que esta sea indispensable para la sociedad actual. Por poner un ejemplo, probablemente el más popular de estos servicios sea la *World Wide Web*, el cual es un servicio de distribución de hipertexto que utiliza Internet como medio de transmisión. La mayoría de estos servicios son de tipo cliente-servidor, lo cual significa que los dispositivos que requieren cierto servicio (clientes) establecen conexión con otros dispositivos que responden a la petición de estos (servidores), siendo toda esta comunicación realizada mediante paquetes de que contiene los datos que se quieren comunicar. A esta arquitectura se la llama cliente-servidor, y es esencial en la arquitectura de Internet.

2.2. Tomografía de la red

Internet, al estar constituida por muchísimas redes, presenta continuas variaciones en su topología causadas principalmente por dos motivos: el primero es que el protocolo TCP/IP es tolerante a fallas, lo cual implica que ante la presencia de fallas el protocolo sigue siendo capaz de entregar los paquetes de datos al destinatario. La otra causa es que al ser operada de forma autónoma por distintas empresas, las mismas pueden introducir cambios en su propia red sin justificar tales cambios ante ninguna entidad.

Por lo ya mencionado en la sección anterior es importantes poder monitorear el estado de la red y así conocer y modelar sus características. Sin esta información resulta imposible desarrollar nuevas tecnologías, protocolos y aplicaciones dado que no se podría conocer ni prever el comportamiento de estos desarrollos al momento de funcionar en entornos reales sobre Internet. Es acá donde entra en juego el concepto de *tomografía de red*, o en nuestro caso más concretamente tomografía de Internet, en el que hacemos referencia al estudio de las características de la red mediante mediciones externas. Estas reciben el nombre de *externas* por ser realizadas siempre entre nodos de la red sin estudiar lo que sucede con los paquetes de datos mientras estos viajan. Para ejemplificar pensemos en el caso de la sección anterior en que un paquete viajaba desde un nodo en USA hasta otro en Nueva Zelanda a través de distintos enlaces de la red. Nosotros mediremos las características del mismo a partir de mediciones realizadas desde nodos remotos, sin estar en contacto directo con cada uno de estos enlaces a través de herramientas que explicaremos más adelante.

Existen dos pasos esenciales en todos los trabajos del tema: la *etapa de exploración* en la que se adquieren por ejemplo datos de los enlaces, las rutas o tráfico; y el proceso de *resolución de aliases* que consiste en el mapeo de direcciones IP previamente halladas en la etapa de exploración a *routers*.

Etapa de exploración

Dentro de la etapa de exploración las técnicas utilizadas se basan principalmente en los conocidos `ping` y `traceroute`. Estas herramientas nos permiten conocer diferentes características de las rutas que siguen los paquetes a través de la red. Para este trabajo en particular haremos amplio uso de los `traceroutes`, de los cuales abordaremos la explicación de su funcionamiento en el siguiente capitulo.

Etapa de resolución de aliases

Como se ha mencionado previamente, los *routers* poseen diferentes interfaces, cada una de las cuales generalmente está conectada a una red distinta y posee su propia dirección IP. Al realizar una exploración mediante `traceroutes`, como se hará en este trabajo, gran parte de

las rutas que se obtengan atravesaran *routers* comunes a través de diferentes interfaces, por lo que es de interés conocer cuales IPs halladas pertenecen a cada uno de estos con el fin de generar un mapa de la topología a un nivel mayor que el obtenido simplemente visualizando las rutas a nivel de IP. Esta tarea de detectar cuales IP pertenecen a un mismo router es lo que se conoce en la bibliografía como *resoluciones de alises*. En esta temática se hace una primera distinción entre técnicas activas y técnicas analíticas (o pasivas) [8]. Las primeras realizan la asociación de IP a *router* enviando sondas (paquetes) a las direcciones a resolver y analizando las respuestas de las mismas. Ejemplo de estas técnicas son `Mercator`, `Ally` y `MIDAR`. Luego, las técnicas pasivas realizan la inferencia sin la necesidad de enviar paquetes a la direcciones en cuestión, en cambio utilizan unicamente la información adquirida en la etapa de exploración para inferir los alias. Ejemplos de estos métodos son `DisCarte` y `Kapar`. En este trabajo se utilizará `MIDAR` [9], la cual será explicada en detalle en el siguiente capitulo.

2.3. Antecedente y proyectos de investigación actuales

Presentada la necesidad de tener conocimiento sobre la estructura de la red, pasaremos a hablar sobre los grupos de investigación y proyectos actuales. En primer lugar debemos mencionar a `CAIDA` [10], la cual es una institución cuya finalidad es generar y recolectar datos de Internet a fin de poder conocer sobre su topología y evolución. El primer proyecto desarrollado en CAIDA fue denominado `Skitter`, en el cual se desarrolló una herramienta capaz de recolectar datos acerca de Internet. Esta herramienta basaba su operación en el comando `traceroute` que, como mencionamos previamente, como resultado de su ejecución devuelve parámetros de la red asociados a una ruta. Luego, a través de las rutas se infieren los parámetros con los cuales se define el comportamiento de Internet. Con esta herramienta se recolectaron datos para un gran número de trabajos de investigación entre los cuales destaca "*Internet tomography*"[11] en donde se sentaron las bases de lo que denominamos *tomografía de Internet*. En este trabajo los autores explicaron que el termino tomografía es útil ya que, al igual que en la practica medica, se obtiene la información de forma poco invasiva. A su vez, se sentaron cuatro ejes imprescindibles según los autores al momento de hacer una relevamiento de las características de Internet. Estos son:

1. Determinar como se interconectan las redes
2. Determinar cuanto demora y por cuantos saltos pasa un paquete entre origen y destino
3. Analizar la frecuencia y el patrón que siguen el cambio de las rutas
4. Visualizar a través de un grafo las forma que toma Internet

Luego del éxito de `Skitter`, CAIDA comenzó un nuevo proyecto denominado `Archipielago` (abreviado como `Ark`) [12]. Este es un proyecto que tiene continuidad hoy en día y en donde se busca relevar datos de Internet a través de la herramienta `paris-traceroute` [13] generando traceroutes desde varios puntos de medición alrededor del planeta. De esta manera se buscó generar una infraestructura capaz de proveer datos en forma periódica y continua con los cueles poder llevar adelante investigaciones sobre la red.

En la actualidad CAIDA lleva adelante investigaciones en múltiples áreas: topología de Internet, protocolos de ruteo, seguridad en redes, análisis de trafico, visualización de redes y relación entre la economía y política con Internet. Una descripción de cada una de estas áreas se puede ver [14].

Dentro del área de topología destaca el proyecto *Macroscopic Topology Measurements*, el cual tiene como objetivo la recolección de datos de conectividad y latencia de todo el espacio de redes IPv4 presentes en Internet, junto con la utilización de esta información para la creación de mapas de Internet con distinto niveles de granularidad (IP, Router, Sistema Autónomo). Las fuentes de información de este proyecto se clasifican en dos tipo: *mediciones activas* y *mediciones pasivas*. Estas ultimas se basan en la obtención de información de las tablas BGP provista por el proyecto `Route Views`, el cual se comentará más adelante en esta misma sección. Las mediciones activas se producen utilizando la infraestructura del proyecto ARK, del cual se habló previamente, mediante la herramienta `Scamper`. Esta permite sondear activamente las rutas IP y medir los `RRTs` desde un host hacia una lista de destinos. Luego, se utiliza `MIDAR` para el proceso de resolución de aliases. Ambas herramientas fueron diseñadas en CAIDA dentro del grupo de herramientas pertenecientes al *Macroscopic Internet Topology Data Kit*. Tanto de `Scamper` como de `MIDAR` hablaremos en el siguiente capitulo describiendo los componentes de estos que son utilizados en nuestro trabajo, no obstante, hay que remarcar que estas presentar más características que las hacen herramientas muy potentes para la realización de diversas clases de exploraciones de la red.

El siguiente proyecto del cual debemos hablar es `DIMES` [15]. Al igual que los proyectos de CAIDA, DIMES surge con el objetivo de determinar métricas para caracterizar Internet tales como la capacidad de enlaces, distancias medidas en cantidad de saltos o latencias. Sin embargo la arquitectura de DIMES varia sustancialmente de la implementada en `Ark`. DIMES se basa en lanzar *traceroutes* en paralelo desde computadoras de usuarios voluntarios que han instalado el software *netDIMES*. Este *software* habilita el uso de la computadora dentro de la plataforma DIMES. Está diseñado de tal forma de servir al proyecto pero al mismo tiempo garantizarle al voluntario la seguridad de su equipo. Además DIMES se compromete a que la utilización del enlace no será mayor a 1KB/s con el objetivo de ser poco invasivo para el usuario y para la red. Este proyecto en la actualidad cuenta con más de 9.600 usuarios distribuidos en todo el mundo.

DIMES no publica toda la información recopilada a través los traceroutes, sino que en lugar de ello publica directamente los enlaces IP hallados. Una diferencia importante a remarcar entre CAIDA y DIMES es que mientras la primera cuanto con la mayoría de sus puntos de medición ubicados en empresas, universidades y entes gubernamentales, los puntos de medición de DIMES se ubican en residencias particulares.

iPlane es un proyecto desarrollado por la Universidad de Washington, con cerca de 250 puntos de medición utilizando la infraestructura de PlanetLab, que permite la construcción de mapas de Internet con el objetivo de predecir el rendimiento de punta a punta entre dos nodos cualesquiera, partiendo de ciertos enlaces cuyo rendimiento es previamente conocido. Típicamente los enlaces conocidos se tratan de enlaces altamente transitados ubicados en la *backbone* de Internet. El objetivo principal de este proyecto es proveer esta información sobre la calidad del enlace minimizando el numero de mediciones requeridas.

El ultimo proyecto que mencionaremos es Route Views [16]. Este es coordinado por la universidad de Oregon y en la actualidad publica diariamente las tablas de ruteo BGP de cientos de *routers* en todo el mundo. Esta tablas son de gran utilidad para múltiples proyectos de investigación de Internet ya que a partir de las mismas de puede deducir gran cantidad de información. Las tablas BGP cuentan con una campo denominado *path* que indica la secuencia de ASN (Numero de sistema autónomo) para alcanzar la red destino. Es decir, a través de las tablas de ruteo se puede ver que inferir la topología de los sistemas autónomos. Relacionado con nuestro trabajo, a partir de esta información se podría generar un grafo a nivel de sistema autónomo de las exploraciones que generemos, no obstante esto ultimo excede el alcance que nos hemos propuesto y queda como una opción de trabajo a futuro.

Capítulo 3

Magallanes

En este capitulo nos dedicaremos a describir `Magallanes`. En primer lugar mostramos cual fue la necesidad que nos llevó a centrar esta tesis en su creación. Luego describiremos el entorno de trabajo en el que opera `Magallanes` y las razones de porqué se ha optado por el mismo. A continuación pasaremos ha hablar propiamente del diseño del programa y la estructura que hemos elegido para almacenar los resultados que nos brinda. Por ultimo hablaremos sobre los problemas con que nos hemos topado en el diseño y las soluciones por que hemos optado.

3.1. Motivación

Con el constante crecimiento de Internet proliferan las técnicas para su estudio en las diferentes temáticas que abarca. Esto proviene de un interés tanto académico como corporativo, gracias al cual hay un trabajo conjunto entre ambos mundos para el constantes desarrollo de nuevas tecnologías junto con el monitoreo y análisis de las ya existentes. En nuestro trabajo nos enfocamos en esto último, en el monitoreo y análisis de las tecnologías existentes y, particularmente, nuestro interés se centra en las exploraciones de Internet. No obstante deberemos abarcar otras áreas que servirán de complemento.

Tal como se viene hablando, el trabajo se enfoca en la temática denominada *Tomografía de Internet*, y más concretamente en el diseño de un programa para la gestión de experimentos cuya finalidad sea el relevamiento de datos y análisis de las características internas de Internet, entre las que se encuentran, por nombrar algunas, la topología, características de los enlaces como ser el ancho de banda o demora, características del tráfico de datos, etc.

Ya se ha hablado de la necesidad de poseer datos actualizados del estado de Internet en forma constante. Debido a la gran cantidad de problemas a afrontar al momento de obtener datos de Internet la comunidad científica no cuenta con una amplia variedad de herramientas disponibles, menos aún personalizables. En cambio, cuenta con un abanico de herramientas

para cada tarea individual, pero que no pueden trabajar por si mismas en conjunto. Esto hace necesario una plataforma con la que se pueda integrar diferentes técnicas, una especie de *Todo en uno*, que simplifique el trabajo de adquisición de datos para el investigador, y además sea escalable para en el futuro poder reemplazar los algoritmos elegidos para ciertas tareas por otros nuevos.

Como se mencionó previamente, el proyecto ARK pone a disposición de la comunidad científica los datos de sus exploraciones [17], pero las mismas no son adaptables a cada necesidad particular. Por poner un ejemplo, si un usuario deseara relevar los datos de latencia entre nodos ubicados en diferentes continentes y un servicio WEB (pudiera ser por ejemplo un servidor de Facebook o Google) con los datos provisto por CAIDA no podría, pero con nuestra plataforma lo haría de una manera muy sencilla, lo cual tiene como incentivo que el investigador pueda dedicarse al análisis de los resultados de las mediciones más que a realizar propiamente dichas mediciones.

En síntesis, el objetivo será proveer a la comunidad científica una herramienta flexible que permita la realización de experimentos personalizables. Debido al gran abanico de posibilidades que esto implica, y al avance continuo en algoritmos y técnicas de análisis, nos propondremos como alcance dejar una plataforma que cumpla con lo hablado, y que además sea fácilmente escalable para futuras modificaciones.

En síntesis, la filosofía que hemos seguido al momento del diseño de `Magallanes` se resume en lo siguiente:

1. **Todo en uno:** desde una única plataforma se puede gestionar todos los pasos necesarios para la adquisición de la información deseada, esto es, agregar nodos, instalación de software requerido en los mismos, generación de las exploraciones, resolución de aliases, etc. Todo esto sin necesidad de recurrir a programas externos.

2. **Modular:** se pueden agregar nuevas funcionalidades sin realizar grandes cambios en el código principal. Cualquier persona puede hacer un *fork* del repositorio donde se aloja el programa y realizar las modificaciones que se ajusten a sus necesidades.

3. **Configurable:** cada exploración es configurada en forma personalizada. Dentro de los parámetros configurables estarán, entre otros, duración de la exploración, cantidad de nodos origen y destino, tipo de traceroute, periodo de traceroutes, etc.

Sin más que agregar, ya pasaremos a la parte más técnica del capitulo.

3.2. Entorno de trabajo

En esta sección comentaremos sobre cada unas de las herramientas que hemos utilizado en el desarrollo de esta tesis. Dada la filosofía del grupo trabajo donde se realizó, la única

restricción que se tiene a priori es que las mismas sean *open source*.

Linux

GNU//Linux es un sistema operativo de código abierto ampliamente utilizado en ámbitos académicos, y en particular en el entorno de trabajo donde se desarrolló esta tesis. Por lo tanto, el software fue diseñado para correr sobre este sistema operativo, y en particular bajo distribuciones basadas en Debian, y no se vio la necesidad de hacer una implementación multi-plataforma.

Python

Python es un lenguaje de programación interpretado de alto nivel, multiparadigma y de código abierto. Lo hemos seleccionado como lenguaje de programación por múltiples motivos. Primero, se trata de un lenguaje con una amplia comunidad de desarrolladores, lo que implica que existan desarrolladas bibliotecas para multitud de tareas, entre las cuales, las necesarias para nuestros propósitos. Entre estos podemos mencionar la posibilidad de interactuar con el sistema operativo y con la base de datos que utilizaremos. A su vez, en el grupo donde se ha llevado a cabo este trabajo se cuenta con una amplia experiencia en su utilización.

Postgres

Dada la necesidad de almacenar grandes volúmenes de datos, y posteriormente acceder a ellos para poder analizarlos y eventualmente procesarlos, requerimos de algún sistema que nos simplifique esta tareas. En este punto entran en juego las bases de datos, en particular hemos optado por una base de datos de tipo relacional por adaptarse a nuestras necesidades sin un elevado grado de complejidad.

Postgres ha sido el motor de base de datos elegido por contar con los requerimientos esenciales de ser *open source*, funcionar en entornos Linux, contar una API para ser operado desde Python y ser de tipo relacional. Adicionalmente, cuanta con características extras que lo hacen ideal para nuestro propósito como poseer tipos de datos nativos orientado al uso en redes (direcciones IP y bloques de direcciones con formato CIDR), y funciones para trabajar con las mismas.

Sobre el diseño de la base de datos que hemos empleado hablaremos en la siguiente sección.

PlanetLab

PlanetLab es una plataforma distribuida geográficamente para el desarrollo, evaluación y acceso a servicios de red a escala planetaria [18]. La misma ha sido diseñada para apoyar el desarrollo de nuevos servicios en redes académicas avanzadas. Este proyecto nace en 2003,

encabezado por la universidad de Princeton, y se lleva a cabo gracias a la suma de un gran número de servidores distribuidos a través de las redes académicas del mundo, las que a su vez, forman un laboratorio computacional a escala global.

Actualmente la misma es mantenida por un grupo de investigadores en 300 sitios pertenecientes a más de 30 países [19], y posee 1.353 nodos en 717 sitios. La ubicación de estos sitios se puede visualizar en la figura 3.1.

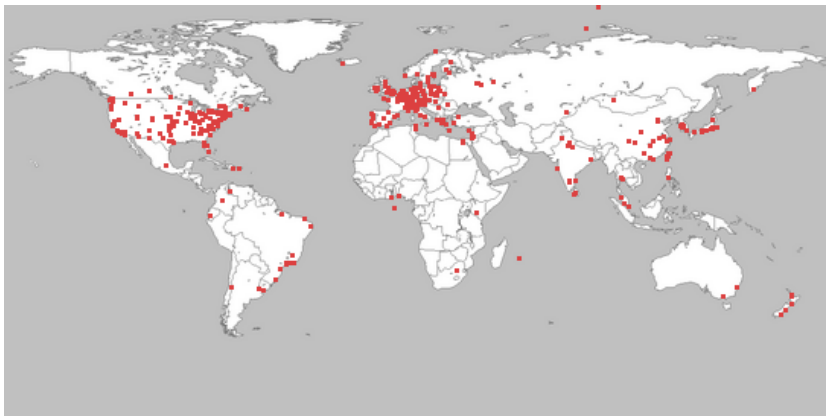


Figura 3.1: Sitios en los que hay nodos de PlanetLab
Fuente: `http://planet-lab.org/generated/World50.png`

En el conjunto de servidores que componen PlanetLab se pueden desarrollar, instalar y ejecutar aplicaciones en un entorno de prueba desplegado sobre una red con condiciones reales. Como se ha dicho, este proyecto nace en 2003, y desde entonces más de 1.000 investigadores de distintas instituciones académicas del mundo y laboratorios de investigación han utilizado PlanetLab para desarrollar nuevas tecnologías para almacenamiento distribuido, mapeo de red, sistemas *peer-to-peer*, tablas *hash* distribuidas y otras aplicaciones.

Pasando a la parte operativa de PlanetLab, a cada uno de los servidores que posee los denominaremos *nodos*. El sistema operativo que utilizan es, en todos los casos, `Fedora Core` de la empresa `Red Hat`. La versión de `Fedora` que utilizan es la 14 -*Laughlin*- para los nodos ubicados en Europa, mientras que la 8 -*Werewolf*- los ubicados en el resto del mundo. La importancia de conocer esto radica en que al ser distribuciones ya antiguas (actualmente la distribución `LTS` es la 21) nos ha traído problema al momento de instalar software y hacer configuraciones del sistema. En la sección siguiente ampliaremos esto y comentaremos como nos hemos arreglado para trabajar con estas distribuciones.

Continuando con la parte operativa, los usuarios de PlanetLab obtienen acceso, debidamente autenticado e independiente de otros usuarios, a los recursos de un nodo en forma de espacios virtuales, los cuales son denominados *slices*. Sobre estos espacios, cada usuario tiene total autonomía. Los nodos son asociados a un *slice*, y sobre estos nodos se crean los espacios virtuales sobre los que luego se pueden instalar las aplicaciones necesarias para correr

diferentes experimentos. Dado que cada espacio virtual funciona de modo seguro y en un ambiente aislado del resto de los espacios virtuales de otros usuarios, el usuario de un *slice* obtiene ciertos privilegios de *root-user* en cada servidor, lo que permite crear nuevos usuarios, controlar los servicios, instalar paquetes, etc. El *slice* con que se ha trabajado a lo largo de esta tesis lo hemos nombrado *uba_conexdat*.

Por ultimo, mencionamos que un usuario por el solo hecho de estar registrado no puede acceder a la creación de estos espacios virtuales, sino que estos deberán ser creados por el responsable de la institución académica, que en el caso de la UBA es el Dr. Ing. Ignacio Alvarez-Hamelin.

Scamper

`Scamper` es una herramienta desarrollada por CAIDA para la realización de diferentes tipos de mediciones en Internet. Dentro estos tipos nos centraremos en los *traceroutes* por tener rol central en la funcionalidad de `Magallanes`.

`Traceroute` es un procedimiento que permite, dentro de ciertos limites, descubrir la ruta que siguen los paquetes IP entre dos equipos. Básicamente, para trazar esta ruta utiliza el campo TTL de la cabecera de los paquetes IP. Dicho campo es un contador que se va decrementando en cada `router` que el paquete atraviesa en la red hasta su destino, y que cuando este llega a un valor 0 es descartado y generado un paquete ICMP tipo 11, *Time Exceeded*, con destino la IP de origen que tenia el paquete descartado. La razón de la existencia de este campo es evitar que ante problemas de paquetes mal ruteados estos queden dando vueltas por la red en forma indefinida. Entonces, como se ilustra en la figura 3.2, `traceroute` utiliza este comportamiento para enviar inicialmente un paquete con $TTL = 1$ de tal modo que el paquete solo llegará al primer *hop* y a continuación espera el paquete ICMP de respuesta. Luego, se envía un nuevo paquete con $TTL = 2$, por lo que atraviesa el primer *hop* y en el segundo salto se agotar el TTL de modo que se genera un nuevo paquete ICMP de respuesta informando del suceso. Este proceso se va repitiendo una y otra vez hasta que se llega al destino o se cumple algún criterio de detención. En el caso de que la respuesta ICMP no se reciba dentro del limite de tiempo configurado, se marca el hop como *no responde*.

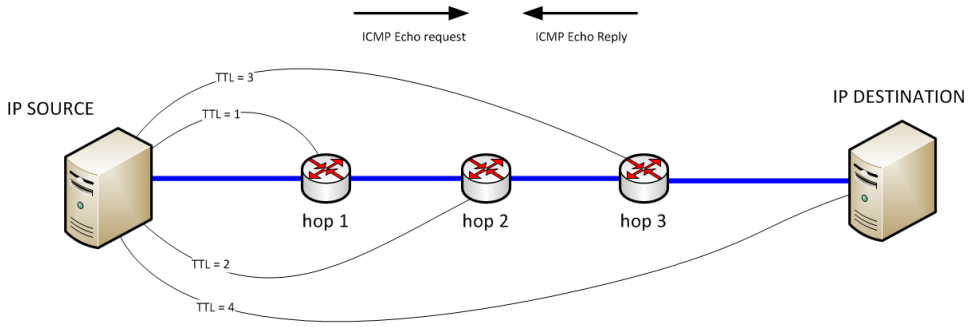


Figura 3.2: Esquema básico de realización de un traceroute

Este procedimiento es muy utilizado para la realización de mediciones de rutas en Internet, y como no hay una especificación que indique como llevarlo a cabo existe gran cantidad de implementaciones del mismo. El problema es que dentro de esta amplia variedad de implementaciones pueden llegar a presentarse problemas metodológicos que provoquen una incorrecta inferencia las rutas, y debido a la importancia que presenta esto en nuestro trabajo explicaremos cuales son los problemas típicos en la inferencia de rutas y como es que se solventan.

Para comenzar a explicar estos problemas introduciremos ciertas características de la operación de los *routers* para luego comentar como es que estas generan problemas en la inferencia de rutas. Supongamos que un *router* tiene tres enlaces, y por uno de estos llega un paquete que para llegar a su destino puede tomar cualquiera de los otros dos enlaces como salida (consideramos que ambas salidas son de igual costo para el protocolo de ruteo presente en la red). Dado que muchos protocolos implementan lo que se conoce como *Balanceo de carga*, una secuencia sucesiva de paquetes podría no seguir siempre el mismo camino dado que el *protocolo de ruteo* puede enviar un paquete por cada salida con el fin de balancear el tráfico en cada una de estas. Este balanceo puede ser implementado por paquete o por flujo, siendo la diferencia de que en el primer caso el *router* envía cada paquete encolado en su *buffer* a través de una interfaz diferente de modo de lograr un balance completo entre sus interfaces, mientras que en el segundo caso se asegura de enviar todos los paquetes de un mismo flujo por una misma interfaz de modo de asegura que los paquetes sean entregados en orden al destinatario. Un flujo de datos queda definido por la quíntupla $\{IP_{origen}, IP_{destino}, Puerto_{origen}, Puerto_{destino}, Protocolo\}$. En este ultimo caso la división de tráfico por cada salida cuando hay un cambio de IP/Puerto en la secuencia de paquetes.

Entonces, definido lo que es el balance de carga, en la figura 3.3 se muestra un diagrama clásico para la explicación de este fenómeno. En el mismo se observa una red que une el nodo *'Src'* con el nodo *'Dst'*. Cuando el primero envía un paquete al segundo, dicho paquete

inicialmente llega al nodo L, y este debe decidir si dirige el paquete por el nodos A o por el nodo B. Suponiendo que el primer paquete es dirigido al nodo A, mientras que el posterior al nodo B, esquematizado en la figura 3.4, al final del `traceroute` estaríamos infiriendo erróneamente la topología de la figura 3.5.

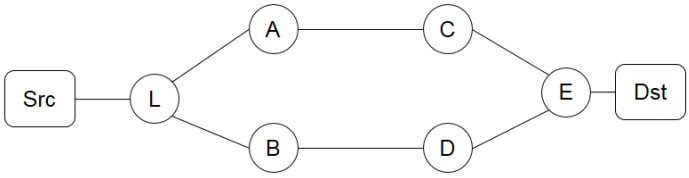


Figura 3.3: Esquema de topología donde el nodo L realiza balance de carga

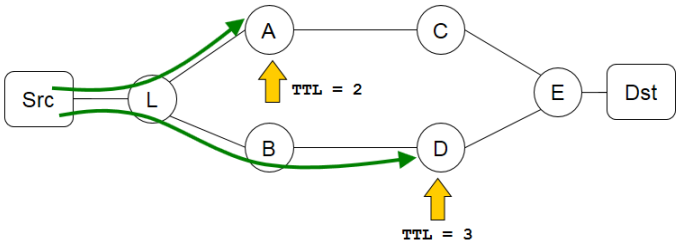


Figura 3.4: Balance de carga en acción

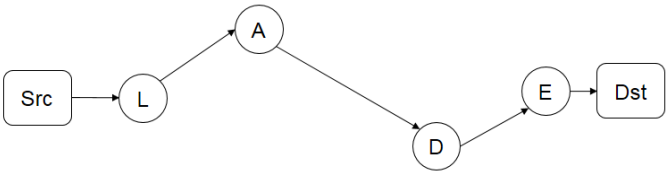


Figura 3.5: Ruta incorrectamente inferida

Demostrado como el balance de carga puede generar inferencia de rutas incorrectas es que entra en juego una implementación de `traceroute` denominada `Paris-Traceroute` [20], la cual permite evitar el balance de carga por flujo mediante la implementación de diversas técnicas y así evitar inferir los falsos enlaces. La clave de esta implementación está en como modifica la cabecera de los paquetes de modo que todos ellos vayan por un mismo camino ante la presencia de un balanceador de carga por flujo. A su vez, permite a los usuarios distinguir entre la presencia de un balanceador por flujo o por paquete. No obstante, dado la naturaleza aleatoria del balanceo por paquete `Paris-Traceroute` no es capaz de enumerar todas las rutas en todas las situaciones, pero sí lo hace considerablemente mejor que la implementación clásica, y puede detectar los casos donde hay incertidumbre.

`Paris-Traceroute` realiza esto modificando ciertos campos, que no son usados por los balanceadores, de la cabecera de los paquetes dentro de los primeros 28 bytes. En el caso de los segmentos TCP se varia el *sequence number*. En el caso de los segmentos UDP el *checksum field*. En este caso se requiere la manipulación del *payload* para lograr el *checksum* deseado, dado que paquetes con un *checksum* incorrecto son descartados. En el caso de paquetes ICMP se configura cuidadosamente los campos *identifier* y *sequence number* para mantener constante la cabecera de todos los paquete hacia un destino a fin de que se conserve valido el *checksum* y los paquetes no sean descartados. En la figura 3.6 se resume el tratamiento realizado a los campos de las cabeceras IP, UDP e ICMP que son usadas por los balanceadores de carga. [21]

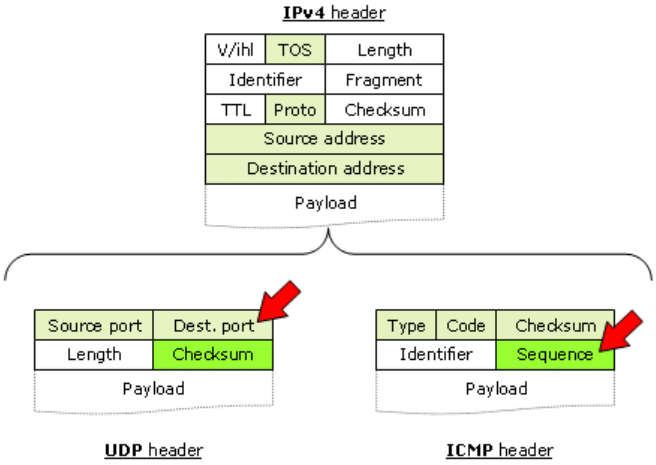


Figura 3.6: Cabeceras IP, UDP e ICMP. El balanceo por flujo utiliza los campos indicados en gris para identificar un flujo. Las flechas rojas muestran los campos incrementados en el implementación clásica de `traceroute`. `Paris-traceroute` utiliza los campos indicado en verde para identificar los paquetes.

Fuente: `https://paris-traceroute.net/images/traceroute_load_balancers.gif`

Esta mejora introducida por `Paris-Traceroute` ha tenido un impacto muy grande al tal punto que ya existen *RFCs*, como la `RFC2991` [22], que recomienda el *balance de carga por flujo* como la técnica más adecuada.

Llegado a este punto debemos aclarar un concepto importante. `Paris-Traceroute` *solo* garantiza evitar el balanceo de carga en los paquetes de ida, pero no así en los paquetes de regreso. Por poner un ejemplo, cuando esta herramienta realiza un `traceroute` sabemos que los paquetes que envía no sufren de balanceo de carga hasta llegar al *hop* destino, pero el paquete de respuesta que este genera puede sufrir balanceo de carga. Este fenómeno se puede observar en la sección *Medición del RTT en los primeros hops*. El `RTT` es el tiempo que

demora en llegar la respuesta de un paquete enviado desde un emisor, es decir, es la suma del tiempo que demora el paquete inicial en llegar al *hop* destino más el tiempo que demora en llegar al nodo origen el paquete de respuesta generado en el *hop*. Entonces, en uno de los cuatro tipos de respuesta características de `RTT` que se han observado al hacer *traceroutes* en los primero cuatro *hops* de una red podemos apreciar que se tienen dos valores dominantes de `RTT`. Como sabemos que al usar `Paris-Traceroute` el camino de ida es siempre el mismo para todo los paquetes podemos inferir que la presencia de dos valores de `RTT` dominantes se debe a que los paquetes de respuesta tienen dos caminos posibles, cada uno de los cuales posee una latencia propia.

El punto final para entender porque hemos utilizado `Scamper` consiste en lo siguiente. Dentro de los proyecto de medición de Internet es necesario realizar múltiples *traceroutes* en simultáneo, debido principalmente al tiempo que demanda obtener la realización de estos. La implementación básica de `Paris-Traceroute` no es capaz de realizar *traceroutes* en paralelo, sin embargo esta restricción la hemos resuelto mediante `Scamper`, dado que este al ser una herramienta específicamente diseñada para el relevamiento de la topología de la red mediante *traceroutes*, incorpora la implementación de `Paris-Traceroute` con las modificaciones necesarios para poder realizarlos en paralelo a múltiples destinos.

GitHub

GitHub es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git [23]. Permite alojar en un repositorio archivos de código y herramientas muy útiles para el trabajo en equipo, como la posibilidad de hacer un *fork* y solicitar *pull request*.

Realizar un *fork* es simplemente clonar un repositorio ajeno (genera una copia en una propia cuenta), para eliminar algún bug o modificar cosas de él. Una vez realizadas las modificaciones se envía un *pull request* al dueño del proyecto. Éste podrá analizar los cambios que se han realizado, y si considera interesante las contribuciones, adjuntarlo con el repositorio original.

Hemos mencionado que GitHub se basa en el sistema de control de versiones denominado `Git`, el cual básicamente es un programa que administra los cambios realizados en un repositorio de código permitiendo que múltiples personas puedan trabajar en simultaneo con él, gestionando los cambios y guardando un historial de los mismos. En este caso, *Git*, es un sistema distribuido de control de versiones, lo que significa que cada persona que trabaje en el proyecto tendrá su copia del mismo, y a medida que se hagan cambios en este, estos se irán propagando a los repositorios locales de los demás desarrolladores.

Finalmente, debemos mencionar que hemos optado por utilizar GitHub para nuestro proyecto dado que contamos con experiencia previa en su utilización, y se adapta perfectamente

a nuestras necesidades. Estas son, principalmente, poder alojar en un repositorio versiones estables del código para su utilización en entornos reales y trabajar con en él desde un *fork* para su desarrollo.

MIDAR

Como hemos explicamos en el capitulo 2, el proceso de resolución de aliases consiste en el mapeos de direcciones IP a *routers*, es decir, dado un grupo de IPs ver cuales de estas pertenecen a las interfaces de un mismo *router*. Este proceso permite obtener un conocimiento de la estructura de la red a nivel de *router*, el cual es esencial para estudiar su comportamiento. En este contexto `MIDAR` [24] es un algoritmo de resolución de aliases desarrollado por CAIDA que implementa varias características que logran hacer que se obtengan resultados más precisos respecto de otros algoritmos junto con la posibilidad de utilizarlo en *targets* de gran tamaño.

`MIDAR` utiliza el hecho de que la mayoría de *routers* poseen un contador común en todas las interfaces con el que generan el valor de identificación de los paquetes que rutean. Dicho campo, ver figura 3.7, posee 16 bits de longitud y es utilizado para identificar paquetes cuando estos se fragmentan en la red. De ahora en más nos referiremos a este valor como IP-ID.

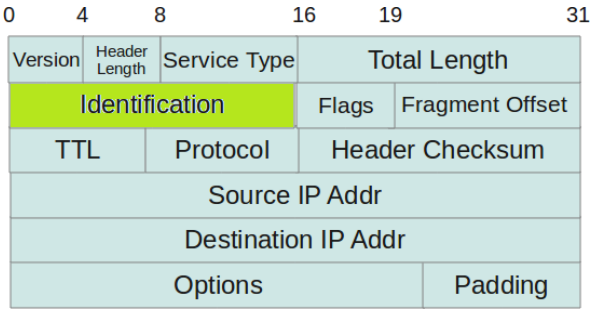


Figura 3.7: Cabecera de un paquete IPv4 con el campo de identificación resaltado

La idea básica detrás de los métodos que utilizan el IP-ID para inferir aliases es que si se envían sondas a dos interfaces pertenecientes a un mismo *router* se podrá detectar si el IP-ID en la respuesta fue generado por el mismo contador y por lo tanto que las dos IP son aliases. Otros algoritmos que trabajan sobre esta idea son `Ally` y `RadarGun`, pero `MIDAR` difiere en como detecta los contadores compartidos, logrando una mejor precisión que las técnicas previas, y además es la única apta para ser usada en producción [9].

Entonces, supongamos que se quiere inferir si dos IP son aliases entre sí. A cada una de estas se le envían sondas con un intervalo de tiempo pequeño entre ambas con el fin de que lleguen al hipotético *router* destino de igual modo con un diferencia de tiempo pequeña y en consecuencia generen dos paquetes de respuesta que tenga un IP-ID similar. Luego, si se

repite este proceso una y otra vez se generarán una series de tiempo como se observa en la figura 3.8.

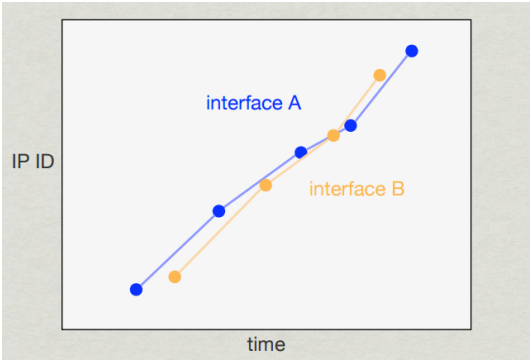


Figura 3.8: Series de tiempo generadas por las sondas enviadas a dos IPs
Fuente: `http://www.caida.org/workshops/isma/1002/slides/aims1002_yhyun_midar.pdf`

Con estas series de tiempo `MIDAR` aplica lo que llama MBT (*monotonic bounds test*) y asume que si dos series de tiempo derivan de un único contador entonces la unión de ambas deben estrictamente monótona creciente. Esto es, ningún IP-ID de respuesta debe ser igual a menor que los previos. Para evitar la incerteza en la medición de tiempos en que la respuestas son generadas en el *router* se utiliza la medición del tiempo entre que la sonda fue enviada y recibida la respuesta. En la figura 3.9 se observa gráficamente como a partir de la serie de tiempo obtenida en una interfaz se puede decidir si la serie de tiempo obtenida de otra interfaz cumple esta condición. En la primer figura se representa la serie de tiempo para una de las interfaces, en la siguiente se muestras rectángulos que representan las zona donde se cumple el criterio de que la unión de las series sea monótona creciente y finalmente en la ultima figura un ejemplo de como podría ser la serie de tiempo de segunda interfaz para considerar que cumplen el MBT.

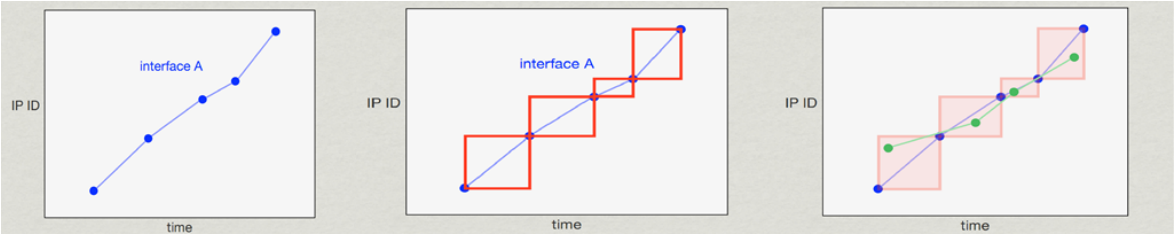


Figura 3.9: Criterio utilizado por el MBT para decidir si dos series de tiempo son monótonas
Fuente: `http://www.caida.org/workshops/isma/1002/slides/aims1002_yhyun_midar.pdf`

Sin embargo, cumplir con el MBT es condición necesaria pero no suficiente para clasificar a las IP como aliases. Esto es, si no se cumplen que las series de tiempo sean monótonas crecientes entonces se puede asegurar que las IP no son alias, pero de cumplir no podemos

asegurar que lo sean. Realizado posteriormente este sondeo una y otra vez se van eliminando falsos-positivos una y otra vez, sin eliminar en ningún caso verdaderos-positivos. Esto hace que luego de una series de comprobaciones se tenga una lista de candidatos a ser aliases con una proporción entre verdaderos-positivos a falsos-positivos muy alta.

Luego, si dos interfaces comparten un contador, estas tendrán series de tiempo similares y además tendrán velocidades de contador, en *IP-ID/seg*, similares. Esto implica, interfaces que tengan velocidades muy distintas difícilmente compartan un contador. Este hecho se utiliza para al momento de realizar los sondeos implementar lo que se denomina una *ventana deslizante*. Para esto se ordenan las IP en orden descendiente de velocidades con el fin de sondear IPs con velocidades similares juntas, es decir, primero se toman las IP con velocidades mayores y se envían sondas. Cuando termina la ronda se toma otro grupo con velocidades menores y se vuelven a enviar sondas y así sucesivamente. Esto se esquematiza en la figura 3.10.

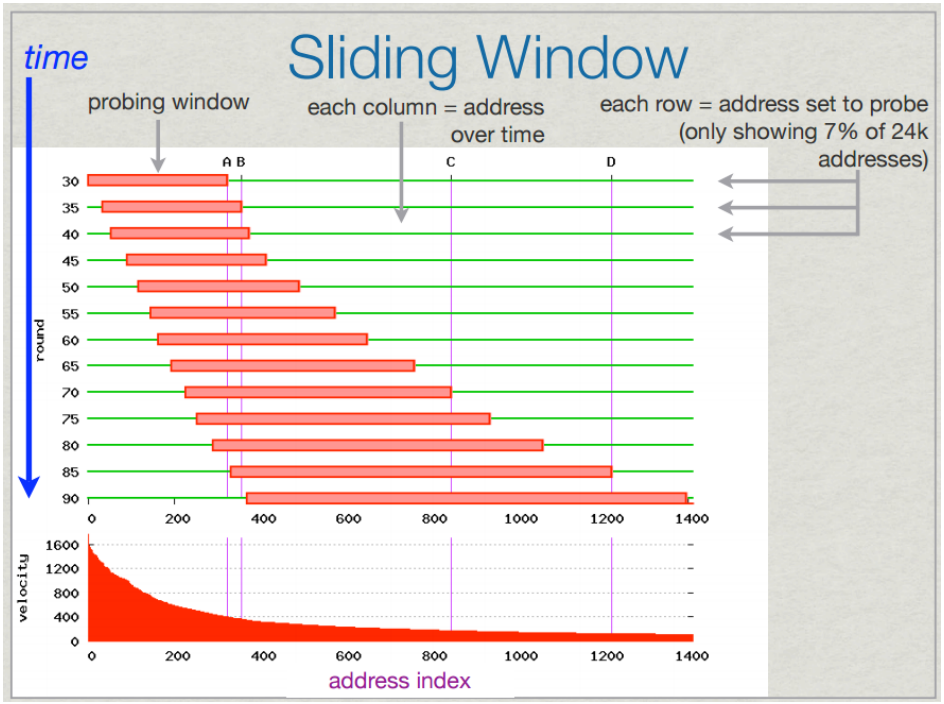


Figura 3.10: Esquema de sondeo mediante ventana deslizante

Fuente: `http://www.caida.org/workshops/isma/1002/slides/aims1002_yhyun_midar.pdf`

Por otra parte, para realizar la comprobación de los IP-ID `MIDAR` implementa cuatro tipo de sondas durante la etapa de estimación de velocidad, y luego elige la mejor para utilizar en las etapas posteriores. Usar múltiples métodos permite recoger series de tiempo para IP que podrían no responder a un tipo particular de sonda. Las sondas utilizadas en cuestión son:

- TCP: envía TCP ACK al puerto 80; recibe con TCP RST.

- UDP: envía UPC a un puedo sin uso; recibe ICMP *port unreachable*.
- ICMP: envía ICMP *Echo Request* a la IP; recibe ICMP *echo reply*.
- Indirect: envía ICMP *echo request* a un *host* pasando el router, pero con un TTL que haga que la sonda expire en este; recibe ICMP *time exceeded*.

Una ejecución de `MIDAR` se dividen en cuatro etapas de sondeo:

- Estimación: para cada IP determina la velocidad de sondeo y el mejor tipo de sonda para usar en las etapas subsiguientes.
- Descubrimiento: sondea todo el target con una ventana deslizante para descubrir todos los pares de direcciones que potencialmente comparten un contador.
- Eliminación: vuelve a sondear todas las potenciales direcciones que pueden ser aliases descubiertas en la etapa anterior para descartar la mayoria de los falsos positivos presentes.
- Corroboración: sondea cada conjunto de potenciales aliases agrupándolas como si fuesen una sola para incrementar grado de confianza en que estas son verdaderos-positivos y descartar los restantes falsos-positivos.

Esto que se ha explicado es una aproximación al funcionamiento de `MIDAR`, se ha evitado hablar de problemas (y soluciones) como el hecho de que los contadores son finitos o que los tiempo en demoran los paquetes en ir y volver a la IP destino no son exactos entre otros. Para una descripción más detallada del funcionamiento de `MIDAR` se puede leer [24].

3.3. Módulos

El diseño de `Magallanes` lo hemos dividido en dos módulos. Por un lado tenemos el modulo que se ocupa de la administración del *slice* y demás tareas de mantenimiento, mientras que por el otro lado tenemos el modulo que se ocupa de la realización de las exploraciones. En esta sección detallaremos ambos, y hablaremos adicionalmente sobre el diseño de base de datos que hemos empleado.

3.3.1. Administración

Desde la *sección de administración* se realizan las acciones necesarias para poner a punto `Magallanes` para comenzar a realizar exploraciones. Como primer punto, para administrar el *slice* de PlanetLab con el que se trabaja contamos con una API que permite, luego de validarse, realizar desde Python las operaciones básica que necesitamos. Dichas operaciones consisten en:

- **Consultar el estado de los nodos:**
  - **Identificadores:** cada nodo tiene asociado dos identificadores, uno que identifica a que institución pertenece (*site_id*), y otro que es único para cada nodo (*node_id*). A su vez, cada nodo tiene sus nombre único en la plataforma (*hostname*).
  - **Estado:** cada nodos puede estar en distintos estados (*boot_state*) según se encuentre operativo, fuera de servicio, en mantenimiento, etc. El único estado en que nos es útil es cuando se encuentra operativo (*boot*).
- **Agregar nodos al *slice*:** mediante el *node_id* se seleccionan los nodos que deseamos agregar al *slice*. Para la selección de los nodos se cuenta con dos posibilidades: 1. Introducir los *node_id* de los nodos que se desea agregar al slice; 2. Agregar nodos aleatorios (en estado *boot*). Es importante remarcar que desde que se agrega un nodo al slice hasta que este es utilizable puede pasar cierto tiempo que depende de cada nodo. En la practica hemos visto que esta demora puede llegar a ser de un día.
- **Quitar nodos del *slice*:** se eliminan los nodo asociados al *slice* seleccionándolos por su *node_id*. También se cuenta con la opción de eliminar todos los nodos que en ese momento no se encuentren en estado *boot* o una cantidad aleatoria de nodos. En este punto es importante remarcar que cuando se elimina un nodo de un *slice*, la instancia dentro de este nodo es eliminada, es decir, se elimina tanto la información como los programas instalados en él.

Luego de haber agregado nodos al *slice* se requiere la instalación de los paquetes y programas necesarios para poder ejecutar las exploraciones. Como muchos de estos paquetes y programas no tienen licencia que nos permita distribuirlo hemos creado un documento que especifica como preparar `Magallanes` para comenzar a utilizarlo luego de descargarlo. Dicho documento se encuentra en el repositorio donde está publicado el software, y lo hemos incluido como el anexo A.

Entonces, una vez preparado el archivo que contiene los programas a distribuir entre los nodos, realizar la instalación resulta muy simple. Basta con seleccionar la opción de instalación deseada y el programa se encarga del resto. Internamente lo que se realiza es la transferencia de un archivos que contienen todos los paquetes y programas requeridos. Una vez completada la transferencia se ejecuta un *shell script* que se encarga de la instalación de cada uno de estos.

Por otro lado, en esta sección también hemos incluido la opción para realizar un *update* a la tabla donde se almacenan los bloques de direcciones IP que se obtienen de `MaxMind` [25]. La utilidad de esta tabla se explica más adelante en este capitulo, en la sección *Selección del target*. Dichos bloques de direcciones se actualizan en la web de `MaxMind` el primer jueves

de cada mes, por lo que si el usuario desea utilizar la ultima versión de los mismo necesita descargarlos desde la web, colocarlos en la carpeta correspondiente, como se indica en el anexo A, y ejecutar la opción correspondiente desde este modulo.

3.3.2. Ejecución de exploraciones

Desde la *sección de exploraciones* se gestionan las exploraciones, es decir, se realizan las siguientes acciones:

- **Programar nuevas exploraciones:** inicialmente se solicita al usuarios la configuración de la exploración que desea correr. Una vez ingresada se crean dos archivos que son transferidos a cada nodo en que se llevará a cabo la exploracion. El primer archivo contienen la lista de IPs que se utilizarán como *target*. La creación de esta lista se explica en *Selección de target*. El siguiente archivo es un *script* escrito en Python que se encarga de gestionar las mediciones en cada nodo según la configuración ingresada. Dentro de este está implementado el algoritmo para *Calculo de First-Hop*, la creación de *logs*, la ejecución de los *traceroutes* y la administración de archivos.

  Referente a la ejecución de `traceroute` cabe mencionar que para poder cumplir con los requisitos de tamaño del *target* se implementó la utilización de múltiples hilos de ejecución. En cada unos de estos hilos se ejecuta una instancia de `Scamper` a la que se asigna una porción del target y una tasa de envío de paquetes (PPS) de modo que la suma de PPS de todos las instancias sea igual a la tasa total.

  Respecto de las opciones de configuración, tenemos las siguientes:

  - Nodos: selección de los nodos origen (monitores) donde se ejecutarán los *traceroutes*.
  - Tipo de target: desde donde y con que criterio se elegirá el target.

    Para correr una exploración exhaustiva, la opción apropiada es *3. Internet; Random* debido a que esta permite seleccionar muchas direcciones IP distribuidas a lo largo del planeta. El criterio de elección de estas IP consiste en tomar una IP aleatoria de cada bloque de direcciones obtenido de la base de datos provista por MaxMind. Una vez que una IP es seleccionada de un bloque, tal bloque no será vuelto a utilizar hasta no acabarse la totalidad de bloques disponibles.
  - Periodo de traceroutes: establece el tiempo mínimo entre las rondas de `traceroutes`, es decir, el tiempo mínimo entre que comienza una ronda de traceroutes y la siguiente.
  - Duración de la exploración: establece la duración de la exploración.

- Tipo de sondas: establece el tipo de `traceroutes` a usar. Los tipos disponibles son: `UDP`, `ICMP`, `UDP-paris`, `ICMP-paris`, `TCP`, y `TCP-ACK`.
- PPS: establece la tasa máxima de paquetes por segundo que utilizará `Scamper` al momento de ejecutar los `traceroutes`.
- GATLIMIT: establece la cantidad de `hops` sin respuesta permitidos hasta comprobar si el nodo destino responde.
- WAIT: establece el tiempo de espera a la respuesta de un `hop`.
- Recalculo de first-hop: Establecer la opción (y el tiempo) de calculo de primer hop.
- Ping: establece si al final de la exploración se realiza una ronda de `ping` a todas las IP halladas en la exploración desde el nodo.
- Descripción: comentario opcional acerca de la exploración.

- **Cancelar exploraciones en ejecución:** se detienen todos los procesos en los nodos donde se esté ejecutando la exploración y se elimina la información residual.
- **Almacenar exploraciones:** una vez finalizada una exploracion, la información se encuentra distribuida en todos los nodos donde se llevó a cabo la misma. En este paso, en forma secuencial, se va transfiriendo cada archivo de resultados desde cada nodo hasta la computadora local en donde a continuación se lo procesa. Inicialmente el archivo de resultados se encuentra en formato `WARTS`, el cual contiene los resultados de los `traceroute` y *ping* en un formato que no nos es simple de interpretar. Dada esta situación, este archivo se lo convierte mediante la herramienta `sc_warts2json` a formato `JSON`, el cual es muy simple de trabajar desde Python por lo que solo resta seccionarlo según el esquema de tablas que hemos definido para almacenar los datos.
- **Realizar resolución de aliases:** el proceso de resolución de aliases se explica en la sección *Resolución de aliases y creación de grafos*.
- **Eliminar datos:** se eliminan todos los datos vinculados a una exploración ya almacenada en la base de datos. Tener en cuenta que en la base de datos se tiene configurado la eliminación en cadena de todos los datos vinculado a una exploracion, es decir, eliminando el registro en la tabla raíz se desencadena la eliminación de los datos en las tablas hijas haciendo que al final del proceso no quede nada de información de la exploración en cuestión.

3.3.3. Estructura de la base de datos

Como hemos mencionado, se ha optado por un modelo de base de datos relacional para la gestión del programa y el almacenamiento de los datos obtenidos en las exploraciones. Dejando

de lado las tablas de uso interno, en esta sección nos concentraremos en el procedimiento para el almacenado de la información de las exploraciones y la forma en que la hemos estructurado.

A pesar de haber implementado un modelo relacional, el esquema que obtuvimos presenta una estructura claramente jerárquica. Esto resultó así luego de hacer un proceso de normalización de los datos que veíamos que necesitaríamos almacenar junto con una adaptación de estos para facilitar el intercambio de datos con otros trabajos realizados en el laboratorio. A su vez, este modelo nos permitió estructurar la información de manera que resultase eficiente tanto su almacenamiento como su posterior procesamiento. En el figura 3.11 se muestra el esquema del que hablamos.

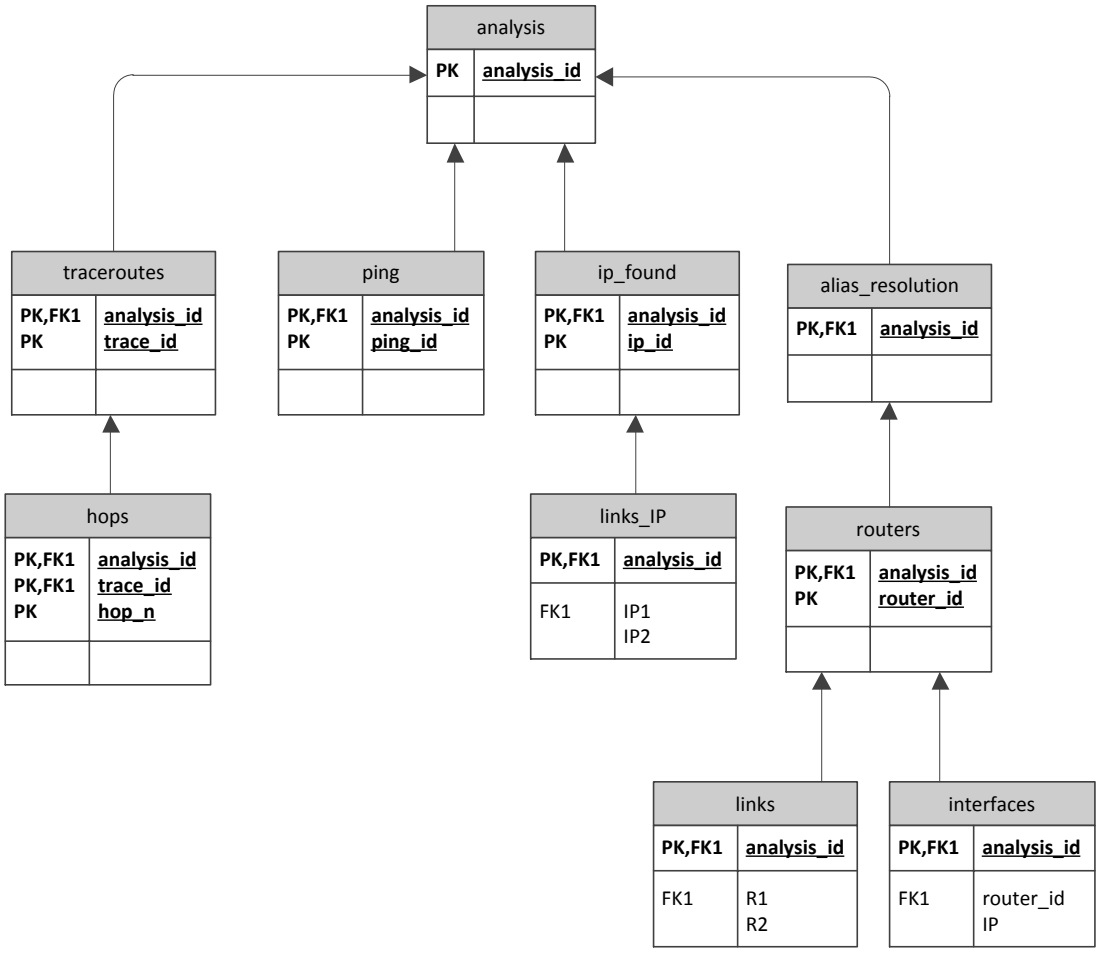


Figura 3.11: Esquema simplificado de la base de datos implementada

Dado que no contamos con un uso especifico para los datos que recolectaremos, sino que el uso dependerá de cada investigador en particular que utilice la herramienta, nos hemos decidido por almacenar todos los datos que nos son provistos por las diferentes herramientas utilizadas.

Como se observa en la jerarquía, la tabla raíz la hemos denominado ANALYSIS. Esta contiene los datos más generales de la exploración y permitiría la reproducción de la misma en otro momento. Luego tenemos las tablas TRACEROUTES y HOPS, las cuales contienen la información de los *traceroutes*. En la primera se almacenan las características de estos, mientras que en la segunda los resultados. Continuando, tenemos la tabla PING donde se almacenan los resultados de los `ping` en caso de programarse su ejecución. La tabla IP_FOUND e IP_LINKS contienen las IP halladas en la exploración y los enlaces entre estos respectivamente, es decir, con estas dos tablas tenemos el grafo de la red a nivel IP hallado en la exploración. Por ultimo tenemos las tablas vinculadas a la resolución de aliases. La primera es ALIAS_RESOLUTION y contiene la información general del proceso, mientras que ROUTER, LINKS e INTERFACES contienen los resultados del proceso, y de estas de puede extraer el grafo de la red a nivel *router*.

Por ultimo, para lograr la integridad de la información hemos colocado en cada una de estas tablas las correspondientes llaves primarias y foráneas, así como indices que nos optimizan las búsquedas. A su vez hemos programado la eliminación en cascada de la información para evitar residuos indeseado en caso de eliminar registros.

3.4. Algoritmos

En esta sección detallaremos algunos de los algoritmos más relevantes que hemos implementado en `Magallanes`.

3.4.1. Selección de target

Al momento de seleccionar las IP destinos hacia las cuales se realizarán los *traceroutes* hemos optado por incluir cuatro opciones. Las dos primeras corresponder a usar como destino nodos de PlanetLab, tanto específicos como aleatorios, de tal modo de poder ejecutar *traceroutes* desde varios extremos entre si mismos.

La tercera opción corresponder con poder indicar manualmente las IP destino, o la URL de esta, que se desean utilizar.

Finalmente, la ultima opción corresponde a tomar direcciones IP aleatorias. Como la idea es que una exploración sea los más abarcativa posible hemos decidido tratar de utilizar direcciones distribuidas alrededor del planeta. Para esto nos hemos valido de las bases de datos libres GeoLite2 provistas por la empresa MaxMind, las cuales son actualizadas en su web el primer jueves de cada mes. Estas bases de datos cubren todo el espacio de direcciones IPv4, incluyendo todas las direcciones publicas disponibles, con una precisión, según MaxMind, en torno a un 99.8 % a nivel de país. Justamente, como nuestros intereses están centrados en poder tomar muestras de direcciones a nivel de países podemos asumir que los datos con que

se trabajan son confiables.

Estas bases de datos provistas por MaxMind son ofrecidas con distinta granularidad, siendo la que utilizaremos es a nivel de país. Esto lo hemos decidido así dado que nos basta para nuestros propósitos. Entonces, lo que hemos hecho fue lo siguiente. Tomamos las dos tablas que nos facilita MaxMind, una que contienen los bloques IP con los códigos de regiones y otra que contiene códigos de países y realizamos un *join* entre ambas. En resumen, lo que finalmente obtenemos y cargamos en nuestra base de datos es una tabla con los siguientes campos:

- `network:` bloque de direcciones con formato CIDR.
- `contienent_name:` nombre del continente en el que se encuentra el bloque de direcciones.
- `contienent_code:` código ISO del continente en el que se encuentra el bloque de direcciones.
- `country_name:` nombre del pais en el que se encuentra el bloque de direcciones.
- `country_iso_code:` código ISO del pais en el que se encuentra el bloque de direcciones.

Una vez que tenemos esta tabla ya estamos listos para la selección del target. Cuando el usuario en `Magallanes` selecciona la opción de *target* aleatorio en Internet inmediatamente se le pregunta la cantidad de IP a seleccionar y si se usará el mismo *target* en todas los monitores, o se usará uno único en cada nodo. En caso de ir por la primera opción, lo que se realiza es una consulta a la tabla solicitando las redes ordenadas aleatoriamente y luego, secuencialmente, se van tomando una IP de cada bloque por vez. Como normalmente se seleccionan más cantidad de IP que bloques disponibles (en torno a los 180.000), de esta forma nos aseguramos una buena distribución de IPs en el planeta.

Para tener una idea de la distribución de bloques que poseemos, en la tabla 3.1 se muestra la distribución por continente.

Continente	#Prefijo	Proporción
EU	78.433	46.8 %
NA	49.784	29.7 %
AS	23.172	13.8 %
OC	6.631	3.9 %
SA	5.622	3.3 %
AF	3.990	2.3 %
An	18	0.01 %

Tabla 3.1: Distribución de cantidad de bloques de direcciones por continente

Para cerrar esta sección hablaremos sobre como se suelen realizar los mapeos de IP a lugar geográfico. Dado que MaxMind es una empresa privada no se cuenta con los detalles de como realiza esta operación, sin embargo podemos mencionar que en primera instancia lo que se utiliza son los registros regionales de Internet (RIR, del ingles *regional Internet register*), los cuales alojan y distribuyen las direcciones IP a organizaciones localizadas en sus respectivos territorios. Tales registros son [26]:

- American Registry for Internet Numbers (`ARIN`)
- RIPE Network Coordination Centre (`RIPE NCC`)
- Asia-Pacific Network Information Centre (`APNIC`)
- Latin American and Caribbean Internet Address Registry (`LACNIC`)
- African Network Information Centre (`AfriNIC`)

Luego, se suelen utilizar varias fuentes para aumentar la precisión. A nombrar:

- Información provistas por ISPs
- *Data mining* y análisis estadístico de información capturada de usuarios.
- *Data scrubbing* para filtrar o identificar anomalías.

3.4.2. Calculo de *First-Hop*

Un problema que se genera al realizar *traceroutes* desde un monitor a múltiples destinos es que los primeros *routers* en las rutas serán siempre los mismos, por lo que de no tomar ninguna medida se podría presentar un bloqueo de tráfico por parte alguno de estos, que consiste en que cuando un *router* detecta que se le envían demasiados paquetes que terminan en él desde un mismo origen deja de responder.

En la figura 3.12 se puede observar el fenómeno que hacemos referencia. En este caso al realizar una ronda de *traceroutes* desde el monitor 'SCR' hacia cada destino A, B, C, D, los nodos N1 y N2 recibirán más cantidad de paquetes que los demás. En particular, en esta topología se observa en el cuadro 3.2 la cantidad de paquetes que expiran en cada uno de los nodos.

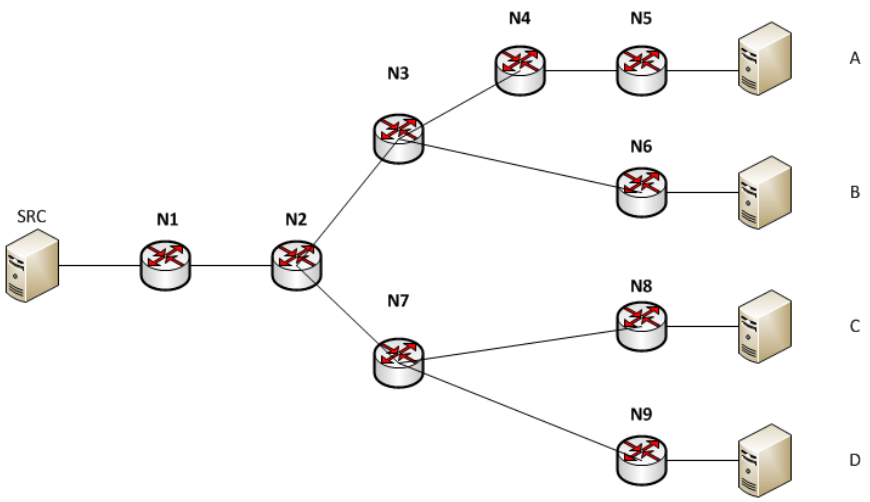


Figura 3.12: Esquema de topología donde el nodo L realiza balance de carga

Nodo	**#Paquetes**
N1	4
N2	4
N3	2
N4	1
N5	1
N6	1
N7	2
N8	1
N9	1

Tabla 3.2: Cantidad de paquetes que recibirán cada nodo en una ronda de traceroutes

El método que implementamos para lidiar con esto consiste en trabajar con el parámetro *first hop* de los *traceroutes*, el cual fija el TTL inicial con que se comenzarán a enviar paquetes. Es decir, si por ejemplo configuramos el *first hop* en 4, el TTL del primer paquete que se envié tendrá $TTL = 4$, por lo que a los tres primeros *hops* no se les estarán enviando paquetes. Definido lo que es el *first hop*, lo que buscamos es que, en una exploración, todo los *routers* que

se descubran tengan balanceado la cantidad de paquetes que se les envían. Mirando la figura 3.12, lo que buscaríamos es que cada *router* de la topología en una ronda de *traceroutes* reciba la misma cantidad de paquetes. Es decir, que luego de la ronda de *traceroutes*, tengamos un único registros por cada IP.

El algoritmo que hemos implementado comienza creando lo que llamamos un *vector de first hop* (linea 1.2), que contendrá todos todas las IP y el *first hop* con el que se configuran los *traceroutes* a cada una de estas: $\{(IP_1, fh_1); (IP_1, fh_1); ...; (IP_n, fh_n)\}$. A partir de este, cuando se configura cada ronda de *traceroutes*, se obtienen todas las IP que se usaran como destino y el parámetro *first hop* de cada una. Inicialmente, en la primer ronda de *traceroutes* (lineas 1.3 a 1.4), se utiliza $TTL = 1$ para cada IP destino, es decir, en la primer ronda de *traceroutes* estaremos en la condición descrita anteriormente en la que cada IP recibe cantidades diferentes de paquetes. Una vez completada la primer ronda de *traceroutes* se hace una actualización a este vector, poniendo en cada *first hop* el valor requerido para lograr que en las sucesivas rondas de *traceroutes* se alcance cada IP una única vez. El valor que deberíamos obtener en el caso de la topología es el indicado en el cuadro 3.3. Si observamos esto valores podemos ver que cada nodo lo alcanzaríamos una única vez de ahora por ronda de traceroute en adelante.

Nodo	**First Hop**
A	1
B	4
C	3
B	4

Tabla 3.3: Cantidad de paquetes que recibirán cada nodo en una ronda de traceroutes

Para hallar los valores del cuadro 3.3 se hace lo siguiente. Se van tomando los *traceroutes* (lineas 5 a 14) de la primer ronda uno a uno y se extrae la menor rama descubierta del árbol que se genera. Entonces, si tomamos el *traceroutes* al destino A tenemos una rama descubierta desde el nodo origen al nodos destino, por lo que el *first hop* para esta IP será 1. Luego, si tomamos como destino el host B, la ruta a este tiene los primeros 3 nodos comunes a la ruta con destino A, y en consecuencia el *first hop* a esta IP tendrá valor 4. Continuando con esta metodología llegamos a la conclusión que para el nodo C requerimos un *first hop* igual a 3, y para el host D un valor de 4. El siguiente pseudocódigo resume el algoritmo implementado:

Algorithm 1 Calculo de *first hop*

Input: *target*: lista de IP a las cuales se harán *traceroutes*

Output: $first\_hop$: vector con el TTL inicial a cada IP del *target*

```
1: IP_found = φ
2: first_hop = φ
3: for i; i++; |target| do
4:     traceroute(IP = IP[i], TTL_initial = 1)
5: for i; i++; |target| do
6:     trace = read_traceroute(IP[i])
7:     TTL_initial = 1
8:     for hop in trace do
9:         if hop in IP_found then
10:            TTL_initial++
11:        else
12:            IP_found append hop
13:            first_hop[i] = TTL_initial
14:            break
15: return first_hop
```

Como las rutas van cambiando durante el tiempo, es de esperar que lo que se halle en un momento difiera de lo hallado tiempo después, por lo que hemos implementado que el usuarios pueda definir cada cuando tiempo desea que se recalcule el *vector de first hop*. Este recalculo consiste simplemente que cuando se cumpla el tiempo definido el *vector de first hop* se reinicia y queda para todas las IP el *first hop* igual a 1 y nuevamente de realiza una secuencia de *traceroutes* para calcular el *first hop* a cada destino. También hemos puesto la opción de desactivar este algoritmo si el usuarios no desea utilizarlo. En tal caso, todas las rondas de *traceroutes* se ejecutaran usando first hop igual a 1 hacia todas las IP destino.

Por otra parte, esta topología es solo un ejemplo simple de lo que podemos encontrar en la red. Si por ejemplo tomamos la topología de la figura 3.3, claramente nos encontraríamos con que entre origen y destino no existe un único camino, no obstante, el algoritmos que hemos implementado ayudaría a minimizar la cantidad de paquetes enviados a cada IP.

Para validar el funcionamiento del algoritmo hemos tomando tres monitores, en cada uno de los cuales ejecutamos dos exploraciones tomando como *target* 100 IPs aleatorias en Internet. La primera exploración la hemos realizado con el algoritmo desactivado (OFF), mientras que en la segunda activando (ON). Una vez completada cada una de las exploraciones extraemos de estas las primeras 50 rondas de *traceroutes*. Como sabemos que en cada ronda se hace un *traceroute* a cada IP del *target*, esto nos da que en total estamos extrayendo 5.000

traceroutes.

Lo que esperamos obtener es que en la primera exploración (algoritmo desactivado) los primeros *hops* sean alcanzados aproximadamente una vez por cada traceroute, es decir, al primer *hop* lo alcanzaríamos en 5.000 oportunidades y a los siguientes *hops* en cantidades similares pero decrecientes dado que luego del primer *hop* las rutas hacia los destinos se van ramificando. Luego, en la exploración que hemos realizado con el algoritmo activado deberíamos alcanzar cada *hops* en un máximo de 149 ocasiones, producto de alcanzarlos 100 veces a cada uno de los *traceroutes* de la ronda inicial, y luego 1 vez en cada una de las próximas 49 rondas.

En la figura 3.13 se han graficado los resultados obtenidos. En cada fila se muestran los resultados en un mismo monitor, siendo en la columna izquierda los datos obtenido con el algoritmo desactivado y en la derecha con el algoritmo activado. Luego, cada barra de los gráficos representa un *hop* alcanzado y la altura de cada barra la cantidad de paquetes que se le ha enviado a cada *hop*.

Lo más importante a observar sobre este análisis es que se cumple lo comentado previamente. El primer caso, con el algoritmo desactivado, vemos como el primer *hop* es alcanzado 5.000 veces y luego los siguientes *hops* en cantidades decrecientes, mientras que con el algoritmo activa la cantidad de veces que alcanzamos cada *hops* es inferior al valor máximo teórico que comentamos previamente (149 veces). En el siguiente caso, tenemos que los cuatro primeros *hops* son alcanzados inicialmente en 5.000 veces, lo cual implica que las rutas a los 100 destinos configurados compartían los estos *hops*. Luego, al activar el algoritmo vemos que todos estos siguen siendo alcanzados en la misma cantidad de veces, pero en una cantidad cercana al valor teórico. Finalmente, el ultimo caso es similar al primero, solo que encontramos que el primer *hop* no ha respondido a los todos los traceroutes lanzados.

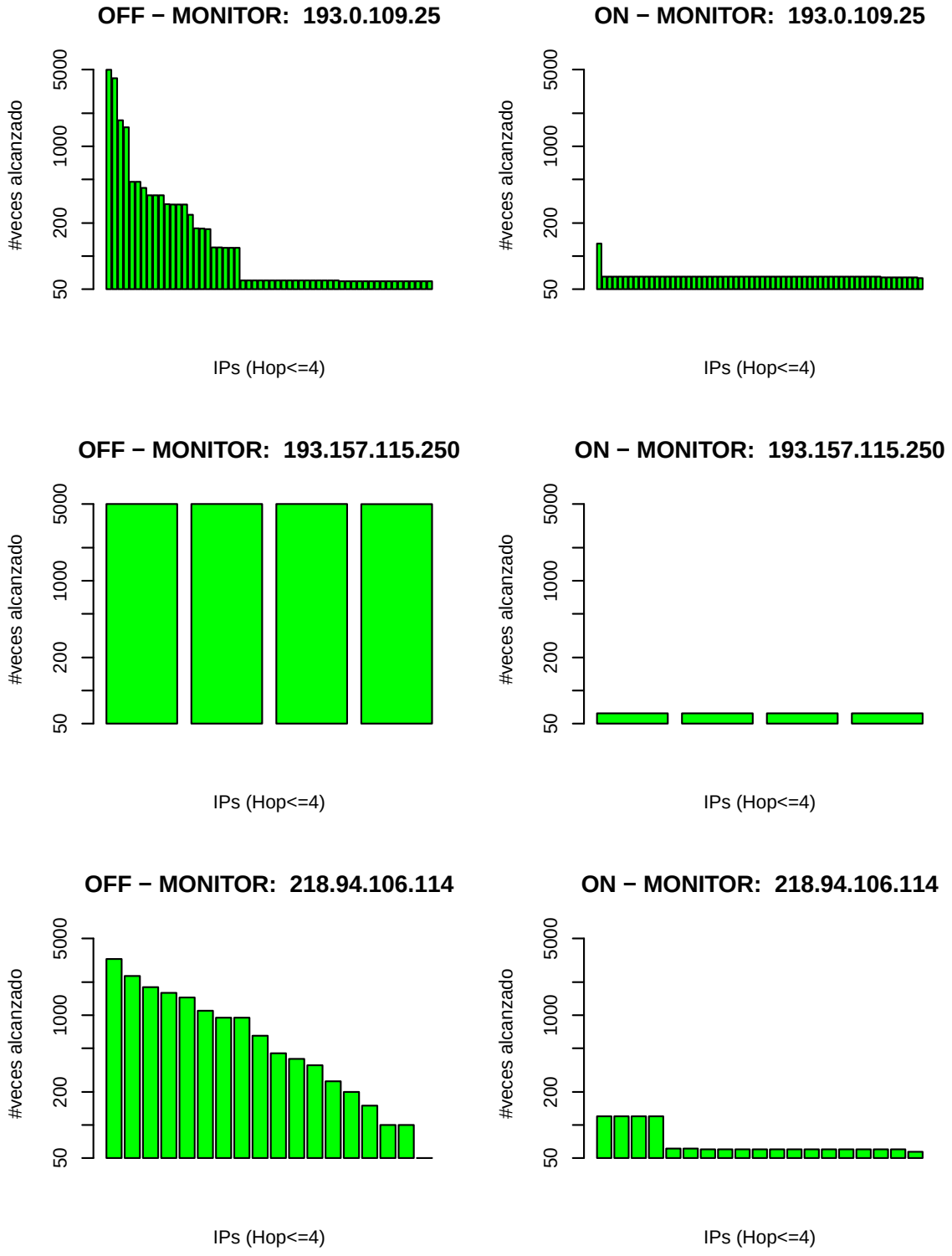


Figura 3.13: Resultados de prueba de funcionamiento del algoritmo de calculo de *first hop*

3.4.3. Resolución de aliases y creación de grafos

Ya hemos explicado previamente en lo que consiste el proceso de resolución de aliases y `MIDAR`, el algoritmo que hemos utilizado en `Magallanes` para esta tarea. Ahora hablaremos sobre como hemos hecho esta implementación. `MIDAR` posee dos modos de uso, uno *local* y otro *distribuido*. En ambos modos de funcionamiento se parte de un *target* compuesto de las IPs que se quieren resolver. En *modo local* lo que hace es enviar desde un único origen todos las sondas a las IP del *target* y luego procesar los datos obtenidos para entregar los resultados. Este modo de funcionamiento tiene la limitante de que el *target* no puede ser superior a las 40.000 IP, lo cual para nuestros propósitos resulta insuficientes. Luego, el *modo distribuido* utiliza múltiples orígenes desde dónde se envían las sondas al *target* dado. Este modo de funcionamiento permite resolver un *target* del orden de millones de IP, y dado que nosotros contamos con múltiples nodos desde donde poder enviar sondas sería lo ideal. Sin embargo, no hemos podido implementar `MIDAR` en forma distribuida dado que para su funcionamiento se requiere la utilización de bibliotecas que no son compatibles con las versiones de los sistemas operativos que utilizan la mayoría de los nodos de PlanetLab.

Ante este panorama de no alcanzarnos con el *target* máximo que permite resolver la implementación en modo local, y la imposibilidad de usar `MIDAR` en modo distribuido hemos tomado una solución mixta. Dicha solución, algoritmo 2, nos fue propuesta en conversación por correo electrónico por Ken Keys, quien es uno de los desarrolladores y responsables del mantenimiento de `MIDAR`.

La misma consiste en tomar el *target* total de IPs halladas en una exploración y dividirlo en grupos de IPs que puedan ser resueltos en forma local, es decir, menores a 40.000 IPs cada uno (linea 2.3). Luego, a cada uno de estos grupos lo pasamos a nodos de PlanetLab que posean `Fedora 14` y ejecutamos `MIDAR` en *modo local* hasta la *fase de estimación*, a fin de obtener para cada IP su *método preferido* de resolución (lineas 2.5 a 2.10). Una vez que conocemos el método de resolución preferido para cada IP generamos nuevamente una división del target, pero esta vez en cada grupo sólo entran IPs que tenga el mismo método preferido (lineas 2.11 a 2.18) y volvemos a ejecutar la `MIDAR` esta vez en forma completa (lineas 2.22 a 2.27). Esta acción de averiguar primero el método de resolución preferido cada IP nos ayuda a minimizar la cantidad de aliases perdidos, dado que en la práctica los autores han descubierto que la cantidad de IP que resultan en aliases poseyendo *método preferido* distinto es pequeño, en torno a un 2 % [24]. Finalmente obtenemos varios conjuntos de *targets* resueltos que transferimos a la base de datos local para continuar con el procesamiento.

El problema de esta implementación es que si ponemos dos IP que pertenecen a un mismo *router*, y por lo tanto el algoritmo debería clasificaras como aliases, en diferentes nodos, dichos aliases los perderíamos. Para minimizar esta perdida de aliases Ken Keys nos ha sugerido que

Algorithm 2 Resolución de aliases implementada en `Magallanes`

Input: R_{IP}: lista de IPs halladas; R_{node}: lista de nodos con `Fedora 14`

Output: aliases

aliases = ϕ
TCP = UDP = ICMP = ϕ
partitioned R_{IP} **in** SR_{IP} subsets / $|SR_{IP}(i)| \leq 40000 \; \forall \; i$
for $i = 1, \; j = 1; \; i++; \; i \leq max(|SR_{IP}|)$ **do**
    repeat
        method($SR_{IP}(i)$, $R_{node}(j)$)
        j++
        if j > j_{max} **then**
            j = 1
    until method $\neq \phi$
    for $n = 1; \; n++; \; n = |SR_{IP}(i)|$ **do**
        select method(n)
        if method(n) = tcp **then**
            insert_ordered(method(n), TCP)
        else if method(n) = udp **then**
            insert_ordered(method(n), UDP)
        else if method(n) = icmp **then**
            insert_ordered(method(n), ICMP)
for each l **in** {TCP, UDP, ICMP} **do**
    select $S_{IP}(i) = \{x \in l / 1^{st}$ *byte identical* **and** $|SR_{IP}(i)| \leq 40000\}$
    j = 1
    repeat
        alias_res($S_{IP}(i)$, $R_{node}(j)$)
        j++
        if j > j_{max} **then**
            j = 1
    until alias_res $\neq \phi$
    aliases $\leftarrow$ aliases + alias_res
return aliases

cuando generemos los *target* a distribuir en los diferentes nodos no lo hagamos aleatoriamente, sino que, aunque no se generen todos los *targets* del mismo tamaño, hagamos una distribución donde todas las IPs con el primer *byte* igual vayan a parar a un mismo *target* (linea 2.20). Esto está relacionado con el hecho de que las IPs no están distribuidas aleatoriamente en el planeta, sino que la mayoría de las IPs contiguas son asignadas a una misma regiones.

Una vez que se ha hecho la resolución de aliases en los nodos, y almacenado lo resultados en la base de datos local, el siguiente paso es la construcción de los grafos a nivel IPs y a nivel *router*. Lo que se tiene hasta el momento es una tabla que contiene todas las IP y el *router* al que pertenecen. En caso de que una IP no se le haya podido encontrar ningún alias consideramos que pertenece a un *router* con una única interfaz. Entonces, el primer paso es hallar los enlaces a nivel de IP, para lo cual consultamos todos los *traceroutes* realizados en la exploración y extraemos todas las IP vecinas, es decir, si en un *traceroute* encontramos que el hop_n corresponde a la IP_1, y el hop_{n+1} corresponde a la IP_2 entonces ambas IP serán vecinas, esto es, existe un enlace entre ambas. Una vez resuelta esto, ya poseemos el grafo a nivel de IP, donde los nodos son las IP y las aristas son los enlaces descubiertos. Continuando, debemos encontrar el grafo a nivel de *router*, para lo cual tomamos cada par de *routers* y definimos que entre ellos existe un enlace si alguna de las IP que lo componen presentan un enlace entre sí. De este modo se ha terminado de generar el grafo a nivel de *router*. A continuación resumimos el procedimiento de creación del grafo:

Algorithm 3 Procedimiento para la creación de grafos

Input: *traceroutes* obtenidos en la exploración

Output: IP_{Links}: Grafo a nivel IP; R_{Links}: Grafo a nivel*router*

1: $IP_{Links} = \phi$
2: $R_{Links} = \phi$
3: **for each** traceroute **do**
4: **for each** hop **do**
5: **if** $IP_{n-1} \neq NULL$ **and** (IP_n, IP_{n-1}) **not in** IP_{Links} **and** (IP_{n-1}, IP_n) **not in** IP_{Links} **then**
6: IP_{Links} **append** (IP_n, IP_{n-1})
7: $R_1 \leftarrow$ **select** router_ID **from** INTERFACES **where** IP $= IP_n$
8: $R_2 \leftarrow$ **select** router_ID **from** INTERFACES **where** IP $= IP_{n-1}$
9: **if** (R_1, R_2) **not in** R_{Links} **and** (R_2, R_1) **not in** R_{Links} **then**
10: R_{Links} **append** (R_1, R_2)
11: **return** IP_{Links}, R_{Links}

Por último, en el siguiente capitulo mostraremos los resultados que hemos obtenido de una exploración en donde hemos hecho la resolución de aliases con esta metodología.

Capítulo 4

Despliegue de experimentos

Ya hemos descrito el funcionamiento de `Magallanes` [1] por lo que ahora pasaremos a ponerlo a prueba con el objetivo de evaluar su desempeño. Comenzaremos realizando una exploración y analizaremos los resultados obtenidos y posteriormente hablaremos de como hemos usado `Magallanes` para la adquisición de datos con el fin de ser utilizados en el desarrollo de un trabajo sobre la influencia de la tecnología MPLS en la topología de Internet. Luego hablaremos de como otros integrantes del laboratorio han empleado `Magallanes` para realizar sus trabajos de investigación sin recurrir al diseño de herramientas propias.

4.1. Exploración de la topología de Internet

En sección realizaremos una exploración con el objetivo de mostrar el funcionamiento de `Magallanes` y, a su vez, extraer información de la estructura de Internet en el momento de ejecución de la misma. Comenzaremos hablando sobre la metodología de trabajo empleada para realizar la exploración, como hemos definido los parámetros de la misma y finalmente haremos un análisis de los resultados obtenidos.

4.1.1. Objetivo y preparación

El objetivo que nos hemos propuesto para la exploración es ser lo más extensa posible a nivel de topología, refiriéndonos a descubrir la mayor cantidad de nodos y enlaces que nos sea posibles, sin focalizar en otros aspectos como la adquisición de datos de latencia y variación sobre los mismos. Esto es debido a que en toda exploración se debe realizar una balance entre los distintos parámetros según el objetivo que se busque. Dichos parámetros en nuestro caso son:

- Cantidad de monitores
- Tamaño del *target* por monitor

- Tiempo entre sucesivas rondas de *traceroutes*
- Duración de la exploración
- Ancho de banda que se desea dedicar
- Espacio disponible para almacenamiento de los resultados

Basándonos en estos parámetros comentaremos como hemos configurado la exploracion, y en caso de haber buscado realizar exploraciones para otros fines indicar que deberíamos haber utilizado.

Luego, lo que buscamos es lograr descubrir la mayor cantidad de nodos y enlaces por lo que en consecuencia debemos usar la mayor cantidad de monitores que sea posible junto el *target* de mayor tamaño que las restricciones que pongamos a la exploración nos permita. Continuando, para maximizar la utilidad de cada monitor se compone cada *target* con grupos disjuntos de IPs, es decir, en cada uno de los monitores se utilizarán IPs únicas en toda la exploración. Cada una de las IP pertenecientes a un mismo target, como hemos explicado en la sección 3.4.1, pertenece a un bloque distinto de direcciones asociado a una región especifica, lo cual nos permite caracterizar la red a nivel planetario. Si en cambio se hubiese querido caracterizar la red de una región en particular, como puede ser Europa, Sudamerica o EEUU hubiera bastado con seleccionar como *target* sólo IPs pertenecientes a bloques asociados a estas regiones. En el caso actual hemos logrado utilizar 104 monitores localizado según se indica en la figura 4.1, y en cada uno se ha configurado un *target* compuesto de 100.000 IPs.

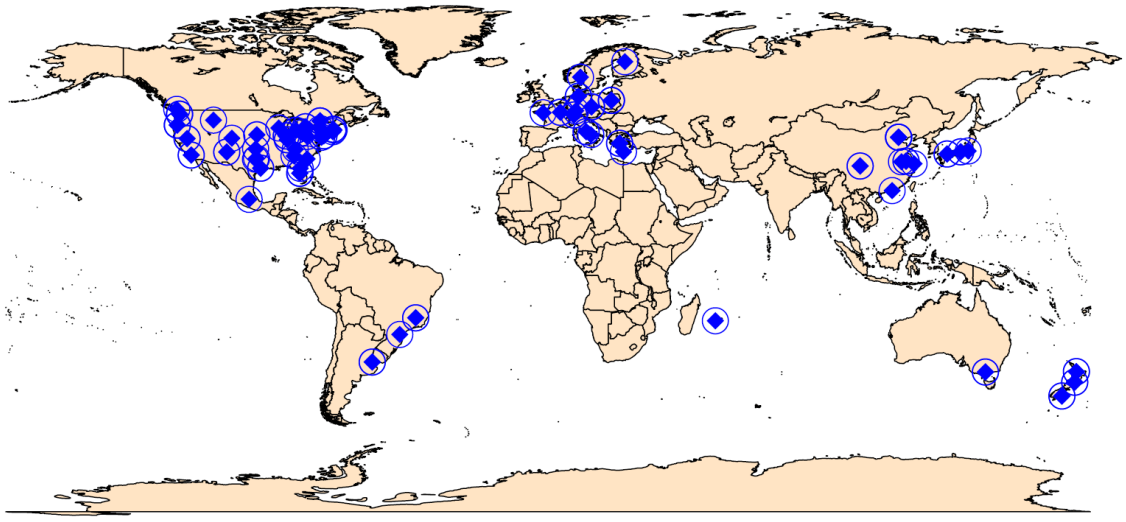


Figura 4.1: Localización de los monitores utilizados en la exploracion

Continuando, dado que realizar una ronda de *traceroutes* (esto es, lanzar un *traceroute* a cada una de las IP del *target*) conlleva un cierto tiempo y ancho de banda, se debe saber

previamente que es lo que se desea caracterizar. Por ejemplo, para un *target* específico a utilizar, si se desea disminuir el tiempo mínimo entre rondas de *traceroute* se necesitaría aumentar el ancho de banda dedicado, o si se busca conservar el ancho da banda y reducir el tiempo entre rondas se requeriría bajar en tamaño del *target*. Para ejemplificar los objetivos, si se busca ver como varían las rutas y/o latencias se debe reducir el tiempo entre sucesivas rondas, ya sea aumentando el ancho de banda o disminuyendo los destinos, mientras que si se busca descubrir la topología no es necesario tener un periodo pequeño sino privilegiar el tamaño del *target*. Este ultimo es justamente nuestro caso, por lo que hemos configurado un periodo de 3600 segundos entre rondas a una tasa de 350 pps (*packets per second*). Para decidir el periodo hemos hecho una estimación diciendo que si tenemos 100.000 IPs a las cuales hacer un *traceroute* por ronda, y suponiendo una distancia media de 10 hops (una cantidad conservadora) requeriríamos enviar 1 millón de paquetes, y como la tasa de envió es 350 pps, esto nos sugiere un estimado de 48 minutos para la ronda.

Finalmente, la duración de una exploración es otro factor a tener en cuenta dado que estas ocupan espacio en disco en los monitores y pueden llegar a suturar a los mismo, además de que cuando se juntan los resultados de cada nodo en la basa de datos central la pueden volver complicada de manejar.

En resumen, nuestra exploración se ha configurado con los siguientes parámetros principales:

- Monitores: 104
- Target: 100.000 IP/monitor (distribuidos en todo el planeta)
- Periodo: 3600 seg
- Duración: 12 horas
- pps: 350
- Tipo de sonda: UDP-paris
- Recalcualo *First-Hop*: Sí (6 horas)
- Resolución de aliases: `MIDAR` (configuración por defecto)

De esta manera esperamos generar idealmente en torno a 125 millones de *traceroutes*.

4.1.2. Resultados experimentales y análisis

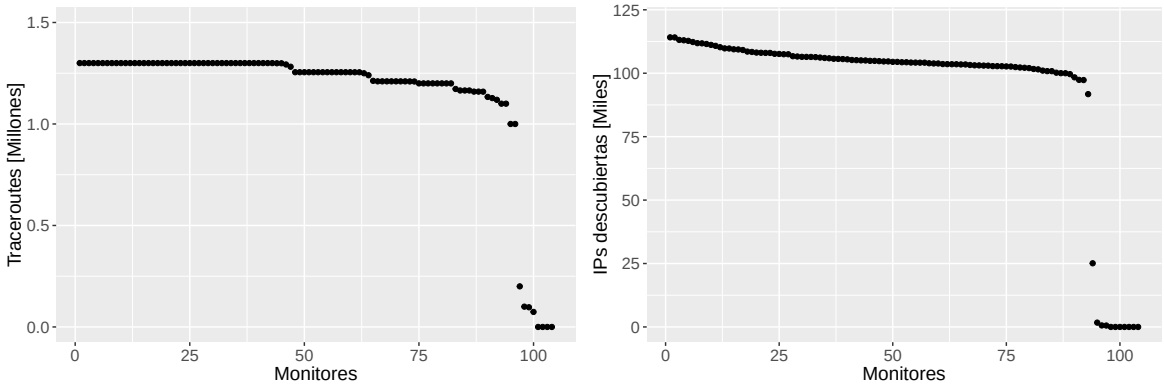
En esta sección presentaremos los resultados obtenidos una vez descargados los datos de cada monitor en la base de datos central. Luego, haremos un breve análisis de los mis-

mos explicando previamente los principales parámetros que son comúnmente utilizados para caracterizar a los grafos durante su análisis.

Resultados

Una vez descargado los datos de los 104 monitores lo primero que se observa es que 4 de estos han fallado, mientras que otros 4 han generado una cantidad de *traceroutes* muy inferior a la esperada. En la figura 4.2a se muestra la cantidad de *traceroutes* registrados por monitor ordenados por cantidad. Dado que los monitores que fallaron representan menos de un 10 % del total no nos detendremos a analizar las causas de estas fallas.

A su vez, la cantidad de IPs descubiertas por monitor individual se muestra en la figura 4.2b. Se observa que tenemos 10 monitores que presentan una cantidad de IPs muy inferior a la mediana. Dentro de estos 10 monitores se encuentran los 8 comentados anteriormente junto a otros 2 que eventualmente presentaran alguna fenómeno que no investigaremos. Finalmente podemos decir que de los 104 nodos donde se corrió la exploración hemos logrado generar resultados correctos en 94 de ellos, lo que nos da una eficiencia superior al 90 % y por lo tanto nos sentimos satisfechos.



(a) #Traceroutes realizados en cada monitor (b) #IPs descubiertas en cada monitor

Figura 4.2: Resultados de la exploración por monitor

Continuando, más allá de tener entre 100 y 125 mil IPs descubiertas en la mayoría de los monitores, es de esperar que la cantidad global de IPs descubiertas sea muy inferior a la suma de estas tomadas individualmente dado que la mayoría de los *traceroute* comparte segmentos de caminos en el centro en la red. La cantidad global de IPs descubiertas es 1.318.872. En la table 4.1 se muestra como se distribuyen estas a nivel continental.

Continente	#IP	Proporción
EU	578.800	43.9 %
NA	378.369	28.7 %
AS	231.092	17.5 %
SA	50.230	3.8 %
OC	26.726	2.0 %
AF	21.577	1.6 %
An	18	0.0 %
no localizables	32.065	2.4 %

Tabla 4.1: Cantidad de IPs halladas por continental

Luego, estas ∼1.3M de IPs están vinculadas mediante 2.351.390 de enlaces. La distribución de los enlaces a nivel de continente se indica en la tabla 4.2.

	AF	SA	OC	AS	NA	EU
EU	5.920	6.629	902	21.722	148.228	977.776
NA	5.179	14.193	5.352	35.590	517.006	148.228
AS	1.277	422	4.374	394.699	35.590	21.722
OC	294	12	34.792	4.374	5.352	902
SA	119	39.077	12	422	14.193	6.629
AF	19.367	119	294	1.277	5.179	5.920

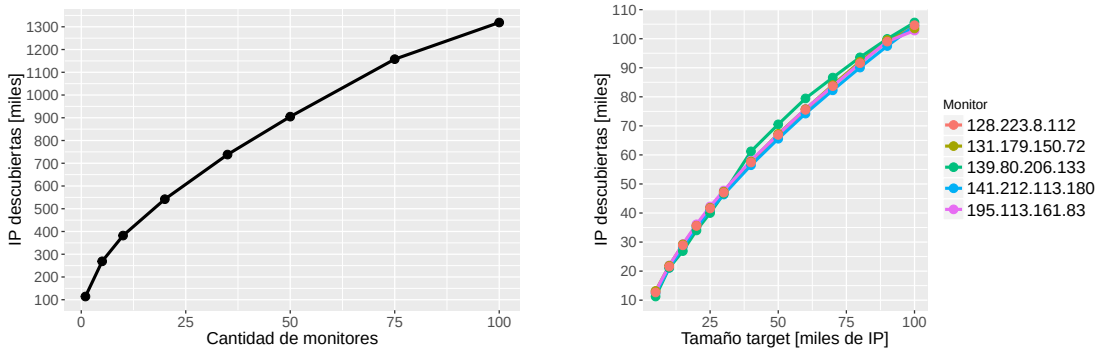
Tabla 4.2: Cantidad de enlaces hallados entre continente a nivel IP

Hemos mencionado previamente que para maximizar el descubrimiento de IPs en la exploración buscamos maximizar tanto la cantidad de monitores como el tamaño del *target* a utilizar.

En la figura 4.3a se observa la relación entre la cantidad de monitores utilizados y la cantidad de IPs únicas descubiertas. En cada punto, los monitores utilizados se han seleccionado en forma aleatoria. Por ejemplo, para obtener la cantidad de IPs halladas utilizando 20 monitores, a estos los hemos tomado aleatoriamente del conjunto total de monitores y luego obtenidos la cantidad de IPs únicas halladas desde estos. Si consideramos por sobre una cantidad mayor a 20 monitores podemos apreciar que, en el rango donde trabajamos, por cada monitor extra en la exploración descubrimos en torno a 10.000 IP nuevas.

Luego, en la figura 4.3b, se tiene el gráfico de IPs descubiertas por tamaño del *target*. Hemos tomado 5 monitores aleatorios cuya cantidad de IPs descubiertas se encuentran en el segundo y tercer cuartil, y observado para distintos tamaños de *target* cuantas IPs se han

descubierto localmente. Se observa que, en el rango de trabajo, no tenemos un decaimiento importante en la cantidad de IPs halladas a medida que aumentamos el *target*.



(a) Globalmente (b) En cada monitor

Figura 4.3: IPs descubiertas

Pasando a la resolución de aliases, hemos seguido el procedimiento explicado en la sección 3.4 con el fin de minimizar la perdidas de aliases. Una vez completado el procedimiento obtenemos que de las 1.3M de IPs se han encontrado 102.606 aliases agrupando en total a 402.562 IPs (31 % de las IP descubiertas). Una vez completado este paso ya poseemos los grafos que caracterizan a la red: el grafo a nivel IP y el grafo a nivel *Router*.

Ahora continuaremos con el análisis del grafo a nivel *router*, describiéndolo mediante sus principales parámetros. A este grafo lo consideraremos de ahora en más como *no dirigido* y no le asociaremos peso a sus enlaces.

Distribución de grado

Dentro de grafos no dirigidos, se define como grado de un nodo a la cantidad de enlaces que este posee. Una caracterización natural de las propiedades estadísticas de los grafos es proporcionada por la distribución de grado de los nodo que lo conforman, lo cual también se puede entender como la probabilidad $P(k)$ de que un nodo aleatorio tenga grado k. De esta manera, la distribución de grado se define según la expresión: $P(k) = \frac{1}{n} \sum_{k_i=k} 1$, donde n es el número de nodos total y k_i el grado del nodo i.

Estudios previos sobre el tema de *redes complejas* han revelado que los grafos muestran una distribución de probabilidad $P(k)$ con cola pesada, la cual en muchos casos puede aproximarse con precisión por una distribución con comportamiento de *ley de potencias*, esto es $P(K) \sim k^{-\gamma}$, donde $2 \leq \gamma \leq 3$. Esto ha llevado a la introducción del concepto de *red de escala libre* [27], las cuales se caracterizan por poseer un pequeño grupo de nodos altamente conectados, mientras que el grado de conexión de los nodos es bastante bajo. Como veremos, Internet presenta este tipo de interacciones entre sus nodos.

En la figura 4.4 se muestra la distribución de grado de $P(k)$ en función de k para el grafo

obtenido en la exploración y como el mismo efectivamente este sigue una ley de potencias. En particular, utilizando la función `power.law.fit{igraph}` de **R**, la cual se basa en el método de Newman [28], se observa que el valor de γ ronda en torno a 2.16 para grado mayor a 5.

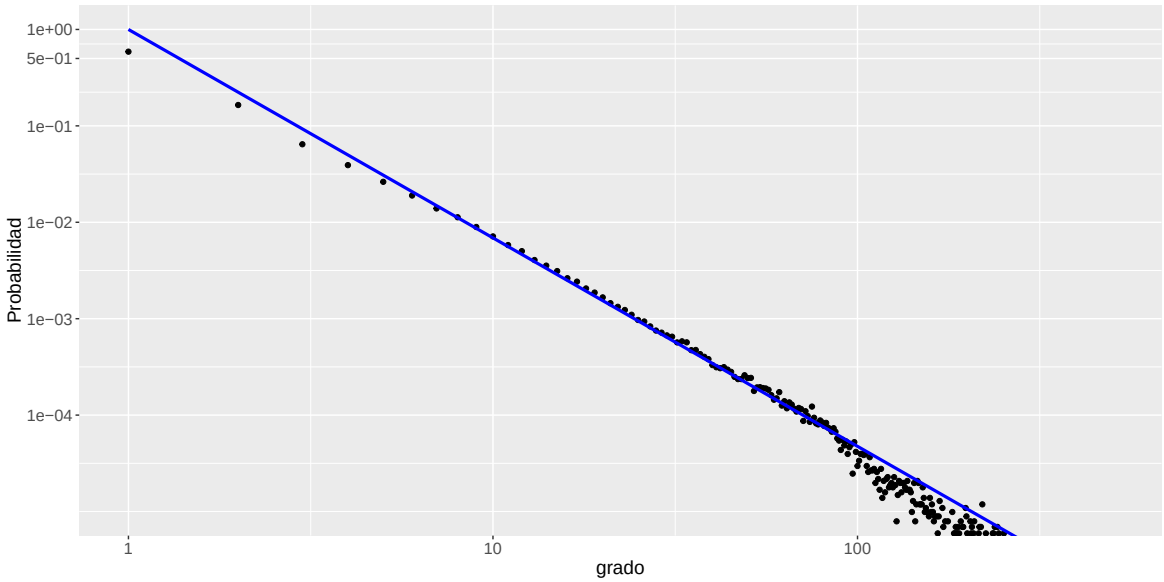


Figura 4.4: Ajuste de distribución de grado mediante $P(k) \sim k^{-2,162}$, para $k_{min} = 5$

Distribución del grado medio de los vecinos

El grado medio de los vecinos de un nodo se define, como su nombre lo indica, como el promedio de grado de los nodos vecinos a este:

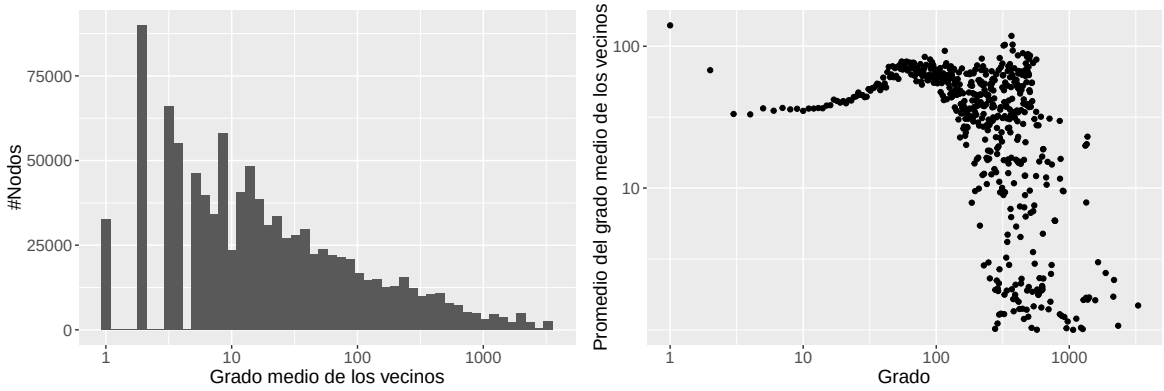
$$k_{nn}(k) = \frac{1}{n_k} \sum_{j/k_j=k} \frac{1}{|V(j)|} \sum_{i \in V(j)} k_i$$

donde $V(j)$ es el conjunto de los vecinos del nodo j, $|V(j)|$ su cardinalidad, k_i es el grado del nodo i, y n_k el numero de nodos de grado k.

Este parámetro da una idea que de como están conectados los nodos de un grafo. En el caso que los nodos de grado elevado posean un k_{nn} pequeño, entonces ellos actuarán de concentradores, implicando que los nodos de grado bajo tengan como vecinos otros de grado elevado. Este comportamiento está definido como *discordante* [29] y se observa en la topología de los sistemas autónomos. En cambio cuando los nodos de grado elevado tienen como vecinos otros nodos de grado también elevado, se denominan *concordante* [29]. Este último comportamiento es observado en algunas redes sociales.

Las nociones de *discordante* o *concordante* tienen sentido cuando las pendientes son marcadas, de otro modo las pequeñas variaciones pueden ser originadas por los sesgos en la obtención de los mapas [30]. En la figura 4.5b se observa justamente este ultimo caso para el mapa de *routers* que hemos hallado, las variaciones son pequeñas (menos de una década

para grado menor que 100). Luego, en la gráfica 4.5a se observa que la mayoría de los nodos están conectados a otros nodos de bajo grado. Esto se explica dada la estructura jerárquica de la red, en la que cada nodo está actúa de puente entre niveles jerárquicos.



(a) Histograma del grado medio de los vecinos

(b) Promedio del grado medio de los vecinos en función del grado

Figura 4.5: Grado medio de los vecinos

Distribución de coeficiente de *clustering*

El coeficiente de clustering C_i de un nodo mide el nivel de interconexión de los vecinos de este, es decir, mide la probabilidad de que los nodos vecinos a este estén conectados entre sí:

$$C_i = \frac{2 \cdot n_{enlace}}{k_i \cdot (k_i - 1)}$$

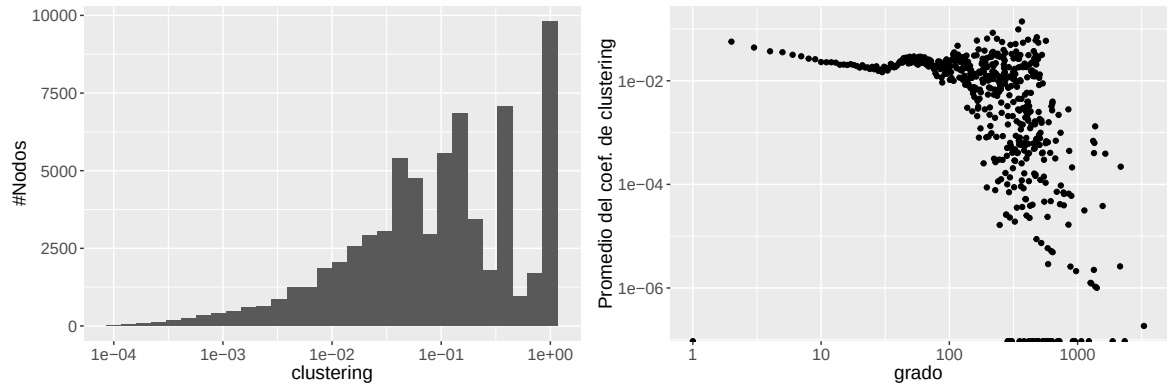
donde n_{enlace} es el numero de conexiones entre los vecinos del nodo i, y k_i es el grado del nodo i. La función de distribución del grado es:

$$C(k) = \frac{1}{n_k} \cdot \sum_{j/k_j=k} C_j$$

Por ejemplo, si el nodo está agrupado como un *clique* su valor es 1 (máximo), mientras que un valor pequeño indica un nodo poco agrupado en la red. Formalmente, el coeficiente de *clustering* de un nodo está dado por la proporción entre los enlaces conectados con sus vecinos respecte del número de enlaces que habría en un *clique* en el que la conectividad es máxima.

En la figura 4.6a se muestra el histograma del coeficiente de clustering del grafo hallado. Se aprecia que en general los nodos tienen un bajo nivel de interconexión en la red, mientras que un grupo tienen un coeficiente cercano a 1. Estos últimos son nodos de bajo grado en los que es más probable encontrar que pertenecen a un clique. En cambio, en los nodos de mayor grados, esto no se da en ningún caso dado la enorme cantidad de enlaces que se necesitarían

para tal hecho. Estos conceptos se visualiza mas claramente en la figura 4.6b, donde vamos que a medida que aumenta el grado de los nodos disminuye el coeficiente de *clustering*.



(a) Histograma de *clustering*

(b) Promedio del coeficiente de clustering en función del grado

Figura 4.6: Coeficiente de *clustering*

Centralidad

La centralidad de un nodo dentro de un grafo se refiere a una medida posible de este en dicho grafo que determina su importancia relativa en la red. Las medidas de centralidad se pueden agrupar en dos categorías: las medidas radiales y mediales. Las primeras toman como punto de referencia un nodo dado que inicia o termina recorridos por la red, mientras que las segundas toman como referencia los recorridos que pasan a través de un nodo dado. Las medidas radiales a su vez se pueden clasificar en medidas de volumen y de longitud, según el tipo de recorridos que consideran. Las primeras miden el volumen (o el número) de recorridos limitados a dicha longitud prefijada, en tanto que las segundas miden la longitud de los recorridos necesarios para alcanzar un volumen prefijado. En base a esta clasificación existen cuatro medidas que son ampliamente utilizadas en análisis de redes:

- **La centralidad de grado (*degree centrality*):**

  Definición: dado un grafo $G := (V, E)$, donde V es su conjunto de vértices y E su conjunto de aristas, entonces para cada nodo $v \in V$ su centralidad de grado $C_{DEG}(v)$ se define como:

  $$C_{DEG}(v) = grado(v)$$

  Si se tiene la matriz de adyacencia del grafo, donde cada posición a_{ij} asume el valor 1 en caso de existir la arista (i, k) y el valor 0 si no existe, entonces la centralidad de grado de cada nodo j se puede definir como:

  $$C_{DEG}(v) = \sum a_{ij}$$

Para grafos dirigidos, se pueden definir dos medidas de centralidad de grado diferentes, correspondientes al grado de entrada o el de salida.

- **La cercanía (*closeness*):**

  Definición: la cercanía $C_{CLO}(i)$ de un nodo i se define como:

  $$C_{CLO}(i) = \sum S_{ij}$$

  donde S es la *matriz de distancias* de la red, es decir, S es la matriz de distancias de la red, es decir, aquella matriz cuyos elementos (i, j) corresponden a la distancia más corta desde el nodo i hasta el nodo j. La utilidad de esta medida es permitir conocer de alguna forma la accesibilidad de un nodo en la red.

- **La intermediación (*betweenness*):**

  Definición: la intermediación $C_{BET}(i)$ de un nodo i en una red se define como [31]:

  $$C_{BET}(i) = \sum_{j,k} \frac{b_{jik}}{b_{jk}}$$

  donde b_{jk} es el numero de caminos más cortos desde el nodo j hasta el nodo k, y b_{jik} el número de caminos más cortos desde j hasta k que pasan a través del nodo i.

- **La centralidad de vector propio (*eigenvector centrality*):**

  Definición: la centralidad de vector propio corresponde al principal autovector λ de la matriz de adyacencia del grafo analizado. Entonces, dado un grafo $G := (V, E)$, donde V es su conjunto de vértices y E su conjunto de aristas, la ecuación de autovectores a resolver es:

  $$Ax = \lambda x$$

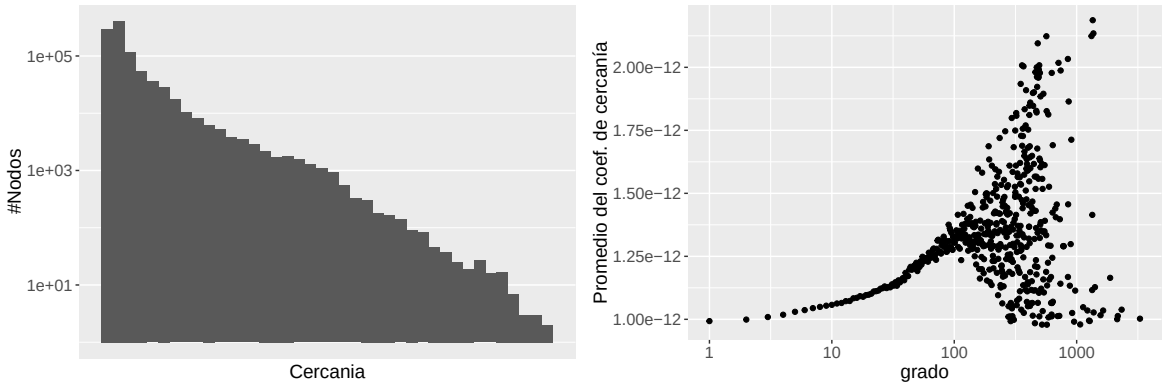
  donde $x(v)$ es la centralidad relativa del vertice v definido como $x_v = \frac{1}{\lambda} \sum_{j \in G} a_{i,j} x_j$.

La primera y la última son medidas radiales de volumen. La segunda es una medida radial de longitud, y la tercera una medida medial.

La centralidad de grado corresponde al número de enlaces que posee un nodo con los demás. Existen criterios que son usados para normalizar esta medida: dividir el grado de cada nodo por el máximo grado obtenido de la red, o bien dividirlo por el número total de nodos de la red. Como esta medida es similar entregaría resultados similares a la distribución de grado que vimos previamente no ahondaremos en la misma [31].

Respecto de la cercanía, dada su definición, se espera que los nodos presentes en el centro de la red (troncal o *backbone*) presenten un nivel de cercanía inferior que los ubicados en las

periferias. En la figura 4.7a se observa el histograma que presenta este parámetro en nuestra exploración. Como es de esperar, dada la forma jerárquica de la red, se obtiene una mayor densidad de nodos con valor bajo de este parámetro. Luego, en la figura 4.7b se observa esta misma característica. A medida que aumenta el grado de los nodos (que en general es propio de los nodos del centro de la red) aumenta el coeficiente de cercanía.

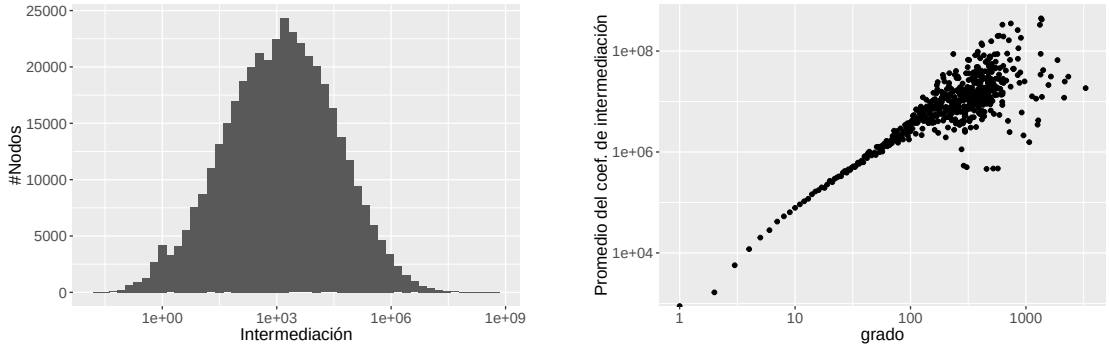


(a) Histograma de cercanía

(b) Promedio del coeficinete de cercanía en función del grado

Figura 4.7: Coeficiente de cercanía

La intermediación de un nodo es una medida que cuantifica la frecuencia o el número de veces que un nodo actúa como un puente a lo largo del camino más corto entre cada par de nodos. Un nodo con un alto grado de intermediación tiene una gran influencia en el tráfico de la red, que en el caso de Internet corresponden a los nodos ubicados en el centro de la red. En la figura 4.8a se observa el histograma de este parámetro utilizando una escala logarítmica sobre el eje horizontal. Luego, en la figura 4.8b se observa esto mismo. A medida que aumenta el grado de los nodos, mayor tiende a ser su coeficiente de intermediación.



(a) Histograma de *intermediación*

(b) Promedio del coeficiente de intermediación en función del grado

Figura 4.8: Coeficiente de intermediación

Por último, la centralidad de vector propio mide influencia de un nodo en una red. Los nodos que poseen un valor alto de esta medida de centralidad están conectados a muchos nodos que a su vez están bien conectados. No abordadores esta medida en este trabajo.

Visualización de topología

Para la visualización del grafo utilizaremos la herramienta `LaNet-vi`, la cual se basa en la descomposición en k-núcleos. Para esto introduciremos algunos conceptos necesarios para describir los resultados obtenidos [32]:

- **k-núcleo:** un subgrafo $H = (C, E|C)$ inducido por el conjunto $C \subseteq V$ en un k-núcleo (o un núcleo de orden k) sii para todo $\nu \in C$: $grado_H(\nu)$ y H es el máximo subgrafo con esta propiedad. Una forma de obtener la descomposición en k-núcleos es ir eliminando recursivamente todos los nodos de grado menor que k, hasta que todos los nodos restantes tengan grado mayor o igual que k.

- **capa:** un nodo tiene número de capa c (*Shell index*) si dicho nodo pertenece al c-núcleo pero no al $(c+1)$-núcleo. Lo importante de este parámetro es su relación con la conectividad. Si bien un k-núcleo no es k-conexo en general, en ciertos casos prácticos se puede dar esta conectividad. Si un grafo es core-conexo, entonces existen k caminos independientes entre todo par de vértices que pertenecen a un k-núcleo. En otras palabras, dado dos vértices u y v, existen al menos $min(c_u, c_v)$ caminos independientes entre ambos. En lenguaje de redes de datos esto significa que si alguno de esos caminos está obstruido, ya sea por congestión o por una falla momentánea, entonces se puede elegir otro camino para llegar al destino. Incluso, si se necesita una red que cumpla con cierta calidad de servicio, es decir que respete ciertas cotas mínimas para algún parámetro (demora, capacidad, etc.), cuanto más caminos existan entre sus vértices, más posibilidades tendrá de encontrar aquel que verifique la calidad deseada.

- **clique:** subgrafo donde cada nodo está conectado a cada uno de los restantes nodos que conforman el subgrafo. Esto es equivalente a decir que el subgrafo es por si mismo un grafo completo.

En la figura 4.10 se muestra el grafo obtenido visualizándolo mediante esta técnica. La idea de la visualización es ubicar en el centro la capa con el K_{max}-núcleo en forma de circulo, donde su diámetro es proporcional al número de elementos que lo componen y luego las otras capas k en circunferencias concéntricas que se alejan del K_{max} y donde la circunferencia exterior corresponde a la capa con K_{min}. De esta manera cada circunferencia corresponde a cada capa k y todos los vértices que la conforman se dibujan sobre ellas con el mismo color. A medida que nos acercamos al núcleo se van solapando algunas de estas circunferencias

por lo cual puede parecer que poseen más de un color, pero realmente son circunferencias solapadas. Si bien los vértices están ubicados inicialmente en una circunferencia, éstos pueden desplazarse hacia el centro según el valor promedio de c_v de sus vecinos v. Esto se observa más notoriamente en las capas externas.

Sobre la derecha de la imagen se muestra una escala de colores con el número de capa c_i, mientras que a la izquierda una escala logarítmica del grado de los vértices representado por el tamaño de los mismos. Los detalles de la visualización se pueden encontrar extensivamente descritos en [33].

En estas visualizaciones se puede apreciar la propiedad jerárquica de Internet, en la cual cada una de las capas está densamente poblada y las conexiones se producen principalmente entre elementos de capas contiguas, y no hacia la capa central como ocurre en otro tipos de redes como la de los sistemas autónomos [30]. Otra observación importante es que no existe fuerte correlación grado-capa. Esto último se puede ver ya que en las capas exteriores tenemos vértices de gran tamaño o grado elevado.

Por ultimo, en la imagen 4.9 hemos puesto la representación que se obtuvo en el trabajo [30] al del mapa de *routers* obtenido por CAIDA para la red en el año 2003. Pasados 13 años desde ese momento hasta la realización de este trabajo se aprecia el gran crecimiento que ha tenido la red. De 32 capas que se obtuvieron en el año 2003 hemos pasado a 54 capas, mientras que el grado máximo alcanzado por los nodos se ha triplicado.

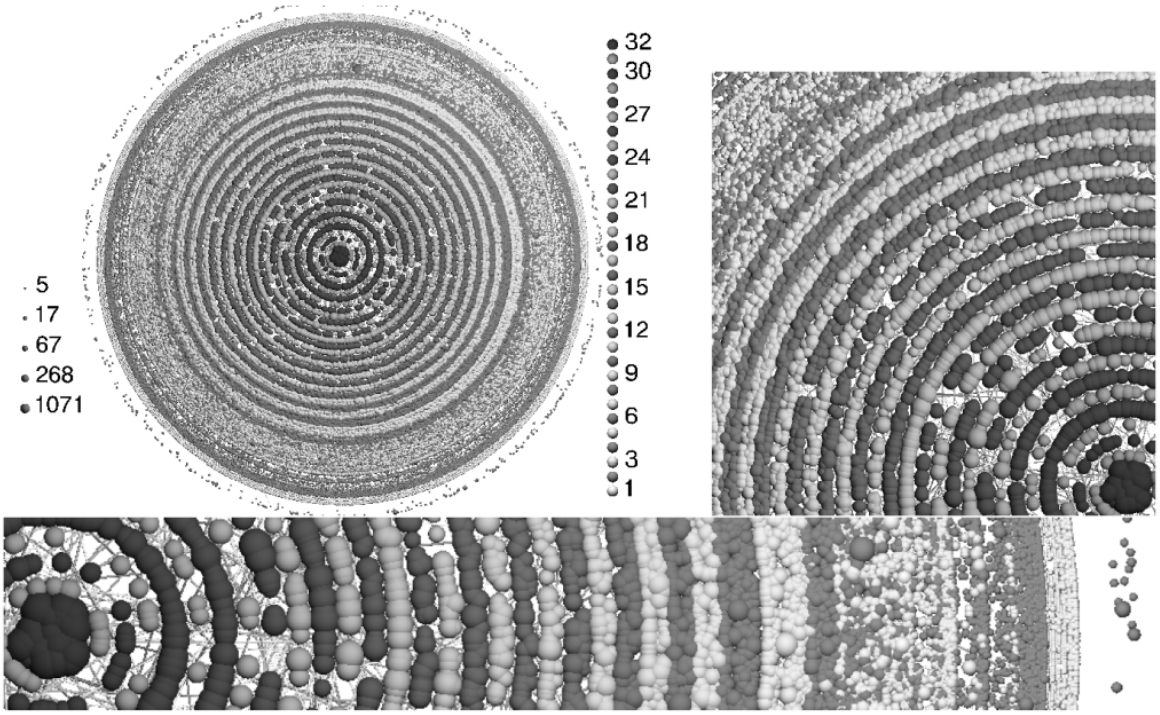
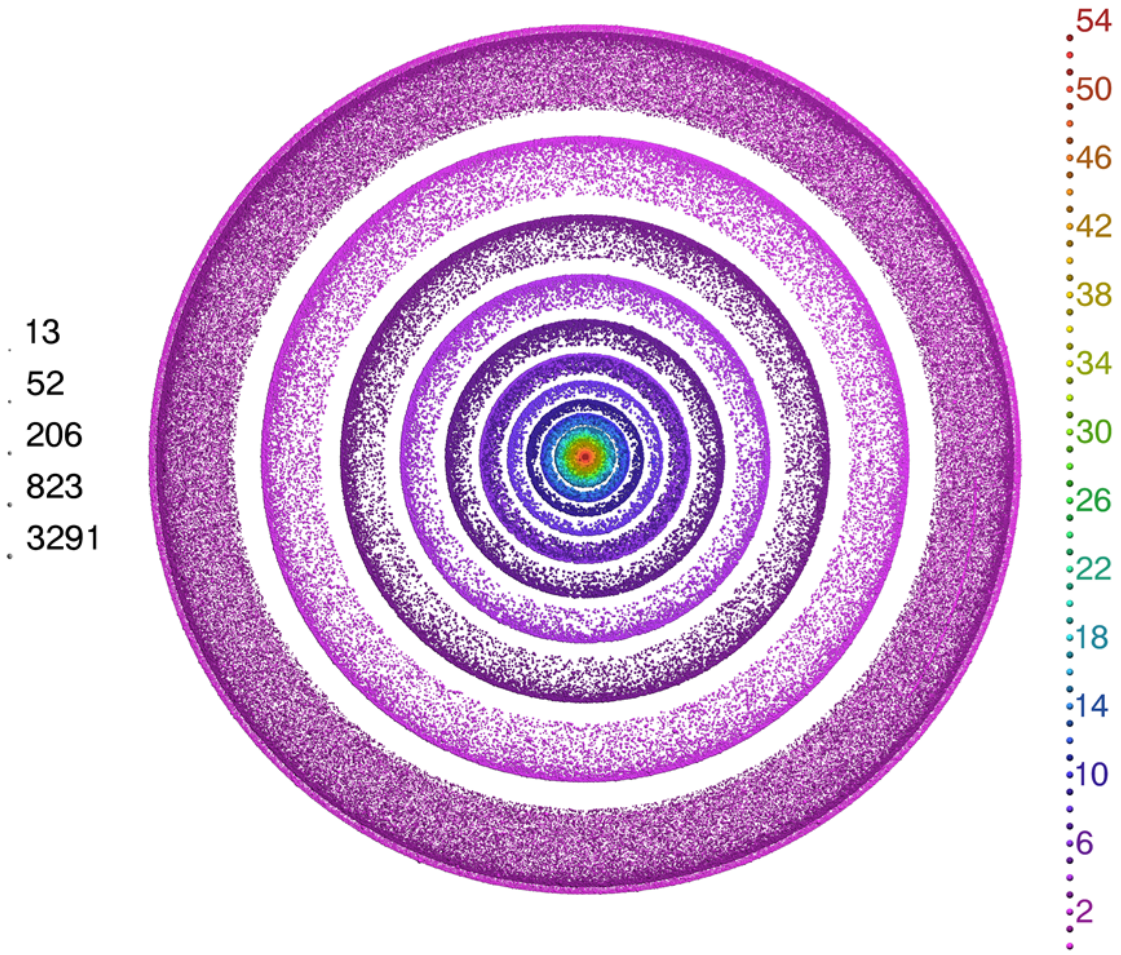
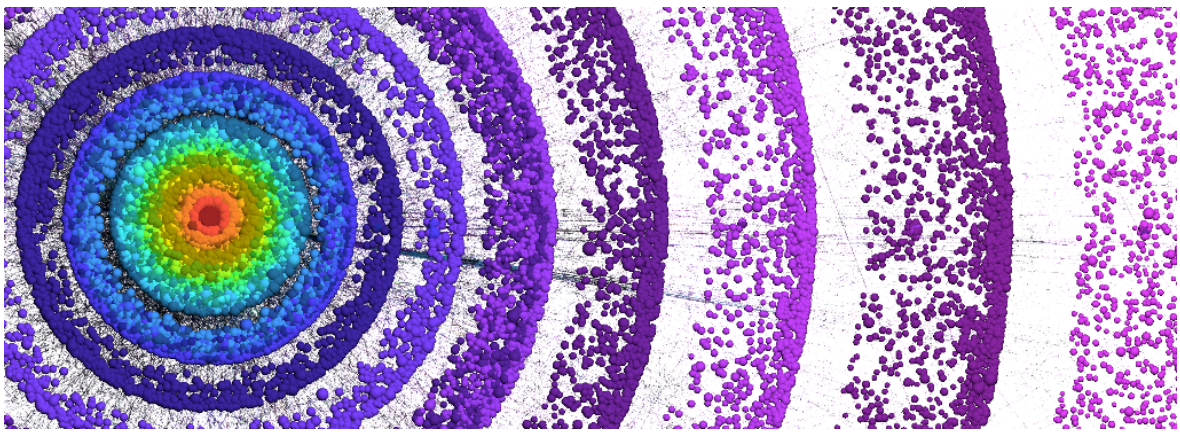


Figura 4.9: Visualización, utilizando la descomposición en k-núcleos, del mapa de *routers* obtenido por CAIDA en el 2003 [30]



(a) Vista completa



(b) Ampliación sobre la zona central del grafo

Figura 4.10: Visualización del grafo obtenido con Lanet-vi

4.2. Revelando la estructura MPLS en la topología de Internet

En esta sección vamos a hablar del trabajo *Unveiling the MPLS Structure on Internet Topology* [2], el cual fue desarrollado dentro del grupo de trabajo `CoNexDat` y presentado en el congreso *8th International Workshop on Traffic Monitoring and Analysis (TMA)*. Para el desarrollo de este trabajo se hecho amplio uso de `Magallanes` con el fin de realizar las mediciones que proveyeron los datos necesarios para el análisis.

Comenzaremos explicando en que consiste la tecnología MPLS y luego pasaremos a explicar los resultados experimentales.

Introducción

La tecnología MPLS (*Multiprotocol Label Switching*) [34] es un protocolo que fue inicialmente desarrollado para reducir el tiempo de envío de paquetes y que actualmente se utiliza principalmente en las redes virtuales `VPN` y en ingeniería de trafico.

La arquitectura MPLS comienza en un router que agrega una o más etiquetas MPLS a cada uno de los paquetes recibidos. Esta etiqueta de 32 bits se agrega antes del encabezado IP y se denomina *label stack entry* (LSE). La misma determina las decisiones que tomarán los siguientes routers capaces de conmutar etiquetas, denominados *Label Switching Router* (LSR). Los caminos formados de las conexiones de varios LSR forman los distintos *Label Switching Path* (LSP) que pueden seguir los paquetes.

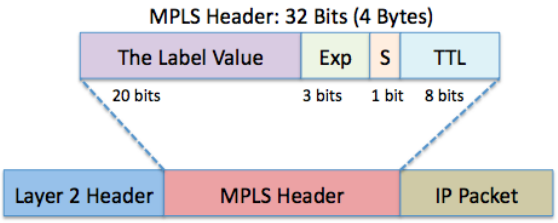


Figura 4.11: Etiqueta MPLS

En una red MPLS los paquetes son enviados analizando la etiqueta de 20 bits del LSE, figura 4.11, la cual también tiene un campo denominado TTL_{lse}. En cada salto MPLS la etiqueta del paquete de entrada es reemplazada por la correspondiente etiqueta de salida. El *fowarding* MPLS es más ligero y rápido que el *fowarding* IP debido a que es más simple hallar la coincidencia para una etiqueta MPLS fija, en lugar que para un prefijo IP en donde es necesario encontrar también la mascara de red adecuada analizando todas las posibles subredes a partir de un prefijo dado.

Los routers MPLS tienen la opción de enviar mensajes *ICMP time-exceeded* cuando el TTL_{lse} del paquete expira como lo hacen los routers IP. También pueden implementar la RFC4950 [35], una extensión de ICMP que permite al router añadir los campos del LSE en el mensaje *ICMP time-exceeded*. Es decir, básicamente copian los campos MPLS del paquete que ha expirado en el mensaje *ICMP time-exceeded*. Actualmente varias versiones de *traceroute*, incluyendo `paris-traceroute` permiten visualizar los valores de la etiqueta MPLS junto al TTL que se muestra por defecto en la salida del *traceroute*, indicando así de manera explicita los LSR atravesados por el *traceroute*. Aún cuando la RFC4950 no este implementada, los LSR que forman un túnel MPLS pueden detectarse si el primer router del LSP (LER de ingreso) copia el valor del TTL_{IP} al campo TTL_{lse}, opción denominada *ttl-propagate*, con lo cual cuando expire el TTL_{lse} cada LSR se anunciará normalmente como un *router* IP mediante mensajes ICMP. Si el TTL_{lse} no vence, su valor se copia nuevamente al campo TTL_{IP} cuando la etiqueta MPLS es removida por el LER de salida.

En ambas situaciones descritas los LSR son visibles al *traceroute*, y si la RFC4950 está implementada se usará la información adicional del mensaje ICMP para identificar los *routers* que forman parte del túnel MPLS, mientras que si solamente está implementada la opción *ttl-propagate* los LSR serán visibles al *traceroute* pero no se podrán identificar como parte de un túnel MPLS.

No obstante, no todos los *routers* están configurados con las opciones *ttl-propagate* o implementan la RFC4950. Si la opción *ttl-propagate* no está habilitada el valor TTL_{IP} no se copia al campo TTL_{lse} y en lugar de ello este se inicializa con un valor aleatorio generalmente cercano a potencia de 2. En este caso el TTL_{lse} no tiene relación alguna con el valor del TTL_{IP}, el cual se disminuirá en uno solamente al salir del túnel MPLS. Bajo estas circunstancias el TTL_{IP} nunca expira dentro del túnel MPLS por lo que los LSR no pueden descubrirse por medio del *traceroute* [36]. Explicadas estas características de los *routers* MPLS, en [37] se propone una taxonomía basada en cuatro clases de túneles MPLS, mostrados en la figura 4.12.

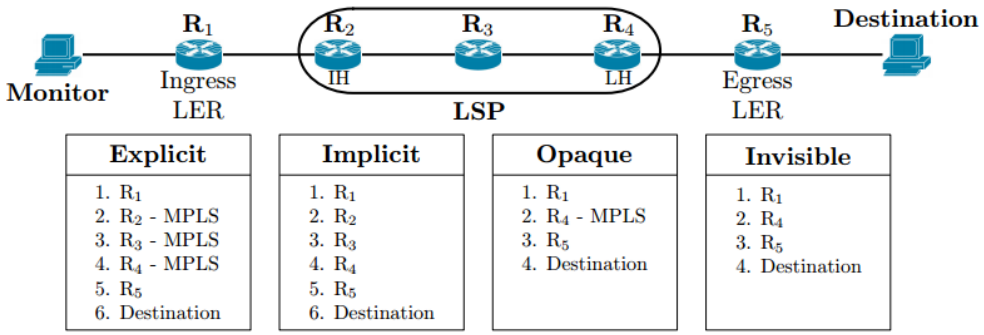


Figura 4.12: Taxonomía de los túneles MPLS

- **Túneles explícitos:** los túneles explícitos se identifican mediante aquellos LSR que tienen habilitadas tanto la opción *ttl-propagate* como la RFC4950. Esto permite que el valor con el cual el TTL_{lse} se inicializa corresponda con el valor que el campo TTL_{IP} tiene al ingresar al túnel MPLS haciendo que todos los LSR sean visibles como si fueran routers IP tradicionales. Por otro lado, los LSR indican de manera explicita en el mensaje *ICMP time-exceeded* que son parte de un túnel MPLS. Se ha encontrado que alrededor del 30 % del total de las rutas descubiertas en base a *traceroutes* tienen al menos un túnel MPLS explicito [36] [37].

- **Túneles Implícitos:** en este caso, los LSR implementan la opción *ttl-propagate* pero no la RFC4950. Esto hace que los LSR sean visibles al traceroute pero en primera instancia se desconozca si un router forma parte o no de un túnel MPLS. En [37] se proponen dos técnicas para inferir si un router pertenece o no a un túnel MPLS, aún cuando esta información no este disponible de manera explicita. La primera técnica se basa en lo que los autores denominan *quoted-TTL* y la segunda en lo que denominan *u-turn signature*.

  - ***Quoted-TTL* (q-ttl):** esta firma se basa en que el valor TTL_{lse} es el que expira dentro del túnel MPLS y no el TTL_{IP}, el cual no varia a partir del momento en que el paquete IP ingresa al túnel MPLS. Por tanto, si se recibe un mensaje *ICMP time-exceeded* a partir de un paquete en el cual $TTL_{IP} > 1$ se puede asegurar que el router que generé dicho mensaje estaba dentro de un túnel MPLS, pues la única posibilidad es que dicho mensaje se haya generado es debido a que expiró el campo TTL_{lse}. Esta información puede ser obtenida analizando los datos del mensaje *ICMP time-exceeded* recibidos como respuesta al *traceroute* ya que en ellos se añade el encabezado IP del mensaje original en donde se incluye el TTL_{IP}

[38]. A este valor del TTL_{IP} extraído del campo de datos del mensaje *ICMP time-exceeded* se le llama *quoted-ttl* (*q-ttl*) y siempre que $q-ttl > 1$ existe un túnel MPLS.

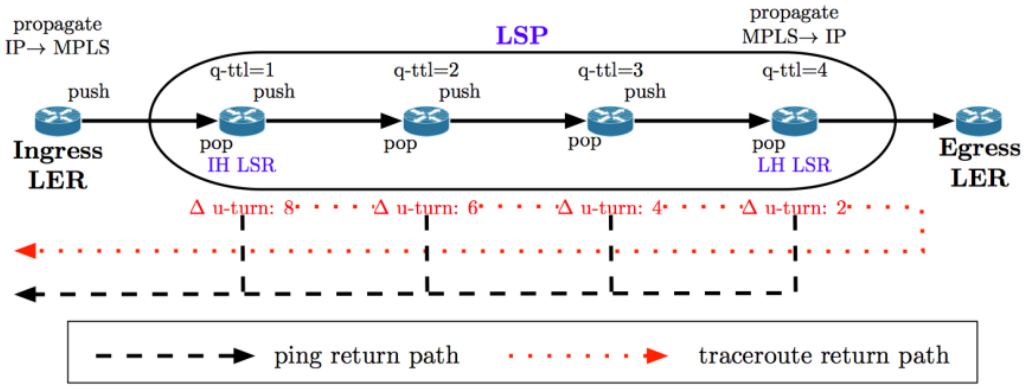


Figura 4.13: Túneles MPLS implícitos identificados por el *q-ttl* y por la firma *u-turn*

El valor *q-ttl* permite también conocer la posición del LSR dentro del túnel como se ilustra en la 4.13.

- ***U-turn signature*:** esta firma se basa en el hecho de que para ciertos fabricantes de *routers*, las configuraciones por defecto permiten que los LSR en un túnel MPLS presenten un comportamiento en común: cuando el TTL_{lse} expira, el *ICMP time-exceeded* generado se envía primero al último hop del LSP y desde ahí se reenvía la respuesta hacia *host* que originó el paquete. Sin embargo, para el resto de paquetes como *ICMP echo*, los LSR son capaces de responder directamente hacia el *host* origen. Esta diferenciación en el comportamiento entre los mensajes *ICMP echo-reply* e *ICMP time-exceeded* permite que se puedan identificar los LSR de un túnel MPLS.

  El método consiste en enviar un *traceroute* que genera un *ICMP time-exceeded* en cada uno de los *hops*, seguido de un *ICMP echo* hacia cada *hop* descubierto por el *traceroute*. En el *host* origen se comparan los resultados recibidos, las diferencias en el TTL de los mensajes determinan la existencia del túnel MPLS. Es decir, si $TTL_{echo-replay} - TTL_{time-exceeded} > 0$ se dice que existe un túnel MPLS y además el valor de la diferencia del TTL varía a lo largo del LSP de la forma $\{X, X-2; X-4; X-6; ...; 2; 0\}$, donde X corresponde a dos veces la longitud del túnel, como se muestra en la figura 4.13.

Se ha encontrado que al rededor del 5.5 % del total de direcciones IP que pertenecen a un LSR se descubren solamente a través de *u-turn signature* o *quoted-ttl* y que los túneles implícitos son 3 veces menos frecuentes que los explícitos [37].

- **Túneles Opacos:** son aquellos en donde el LER de ingreso no tiene habilitada la opción *ttl-propagate* pero el ultimo *router* del LSP tiene implementada la RFC4950. Cuando la opción *ttl-propagate* no esta activada, el LER de ingreso inicializa el TTL_{lse} en 255 haciendo que el paquete nunca expire en el túnel y por tanto todos los LSR se oculten al *traceroute*, excepto el *router* de salida del túnel (LH) que removiendo la etiqueta MPLS vuelve a decremento el TTL_{IP} en lugar del TTL_{lse} como lo hicieron sus predecesores (figura 4.12).

  Cuando el LH recibe el paquete, aún con la etiqueta MPLS, le corresponde un $TTL_{lse} = 255 - (n + 1)$, para un túnel de n *hops*. En [37] se menciona que si esta habilitada la RFC4950, los *routers* forman el mensaje *ICMP time-exceeded* incluyendo la etiqueta MPLS en él, con lo cual se puede determinar en base al TTL_{lse} ($1 < TTL_{lse} < 255$) añadido en el *ICMP time-exceeded*, la existencia de un túnel opaco y su longitud.

  Estudios posteriores, encontraron que la frecuencia de aparición de este tipo de túneles es considerablemente baja (alrededor del 1 % de los túneles MPLS) y que la taxonomía descrita para identificarlos corresponde más a una rara excepción de la regla, en donde la regla la forman los túneles invisibles [39].

- **Túneles Invisibles:** los túneles invisibles no implementan ni la opción *ttl-propagate* ni la RFC4950, haciendo que hasta el momento no exista un estimado de su presencia en Internet.

Desarrollo

Presentada la tecnología MPLS pasaremos al desarrollo en sí del trabajo. El objetivo del mismo fue evaluar las técnicas de detección de túneles presentadas y estudiar el nivel de implementación de esta tecnología en Internet.

Para lo mismo se requería la adquisición de datos de la estructura de la red al momento de llevarse desarrollarse el trabajo (Octubre 2015). Es acá donde entra en juego `Magallanes`, el cual fue utilizado para la adquisición de tales datos. Se ejecutó una exploración utilizando 100 monitores, y un *target* compuesto de 10.000 IPs únicas por cada monitor, aleatoriamente distribuidas en todo el planeta. El tipo de sonda a utilizar fue *ICMP-parís*. Adicionalmente, con el fin de obtener las firmas *u-turn* se configuró para realizar al finalizar la ronda de *traceroutes* una ronda de 6 *ping* (*ICMP echo-replay*) a cada una de las IP halladas durante la exploración. Esta ronda de `ping` tenia como objetivo permitir saber con una confianza del 95 % si hay un único camino de regreso y por lo tanto reducir los errores de medición causados por un caminos de regreso conteniendo balanceadores de carga que ocasionasen que tales caminos sean de distinta longitudes.

Como resultado de esta exploración se descubrieron alrededor de 270 mil IPs, vinculadas

a través de 520 mil enlaces. Estas IP fueron vinculadas a los sistemas autónomos a los que pertenecían mediante el *dataset* de CAIDA derivado de RouteViews y obtenido el mismo día de la exploración. Continuando, en la resolución de aliases se encontró un 19 % de aliases en la topología. Adicionalmente se obtuvo que 44 % de los *traceroutes* contenían al menos una etiqueta MPLS, donde la cantidad de túneles explícitos es altamente superior a los implícitos. Se descubrió en 34 % de los *traceroutes* y al menos un túnel implícito en 16 %. Sorprendentemente, se encontraron mas túneles implícitos mediante la firma *u-turn* (12 %) que mediante la firma *q-ttl* (4 %). Sin embargo, esta ultima coincidió en al menos 63 % de los túneles explícitos.

Con esto datos se paso a la validación de nuestra metodología utilizando las firmas *u-turn* y *q-ttl* para el descubrimiento de túneles implícitos. Básicamente, comparamos la posición del LSR dentro de un túnel MPLS, llamada *posición MPLS*, con los valores de las firmas. El objetivo de esto es poder evaluar la precisión de la firma *u-turn*.

La *posición MPLS* de un LSP es obtenida basada en su orden de apariciones en un LSR. El orden de aparición es llamado *n-posición*, esto es, el primer LSR revelado por un *traceroute* debe ser el primer LSR dentro del LSP (*1-posición*), el LSR revelado luego debe ser el segundo LSR dentro del LSP (*2-posición*), y así. Dado que los túneles MPLS pueden ser configurados para realizar balance de carga, el *n-posición* revelada por *traceroute* podría conducir a un *bias* con respecto a la real *posición MPLS*. Esto es ilustrado en la figura 4.14.

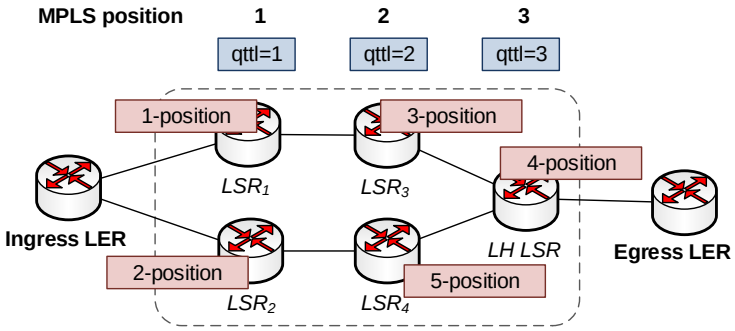
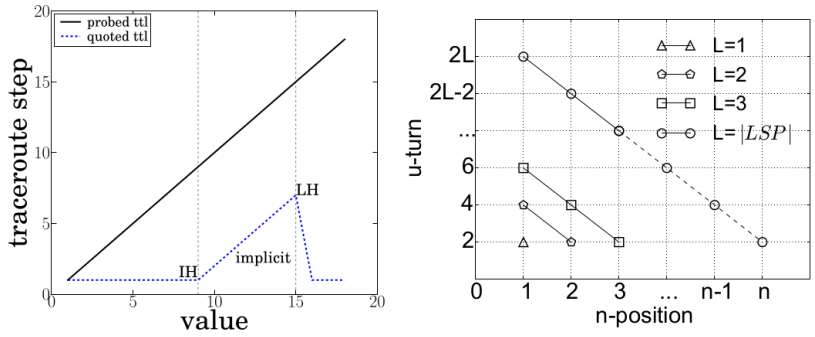


Figura 4.14: Ejemplo de *bias n-position*. Se muestra un escenario donde, debido al balanceo de carga por paquete y a la ausencia de la implementación de la RFC4950 en el LSR_1, la *n-posición* podría ser erróneamente inferida. Primero, *traceroute* revela el ultimo hop del LSR (LH). Finalmente, debido al balanceo de carga en el LSR_2 el LH es revelado dos veces (con diferente valor *q-ttl*).

Los túneles implícitos están basados en la firma *q-ttl* o en la *u-turn*. Ambas están directamente relacionadas con la *posición MPLS*. En efecto, primero, el valor *q-ttl* se refiere al *IP-TTL* del paquete *echo-request* cuando entra en el túnel MPLS. Por lo tanto, un *q-ttl* de

n en el resultado del *ICMP time-exceeded* significa que el paquete enviado expiró en el *hop* n luego de que ingresó, esto es, un LSR que responde con un *q-ttl* $= n$ significa que el LSE aparece en la *n-posición* en el LSP. Esto es ilustrado en la figura 4.15(a). Desde el LER de ingreso, el *q-ttl* comienza a crecer linealmente con la longitud del LSP. Nosotros por tanto esperamos observar un $q - ttl = 1$ en el primer LSR en el LSP, un $q - ttl = 2$ en el segundo y así sucesivamente. Luego, el valor *u-turn* está relacionado con la longitud del túnel (L) y la *n-posición* del LSR dentro del túnel como se muestra en la 4.15(b).



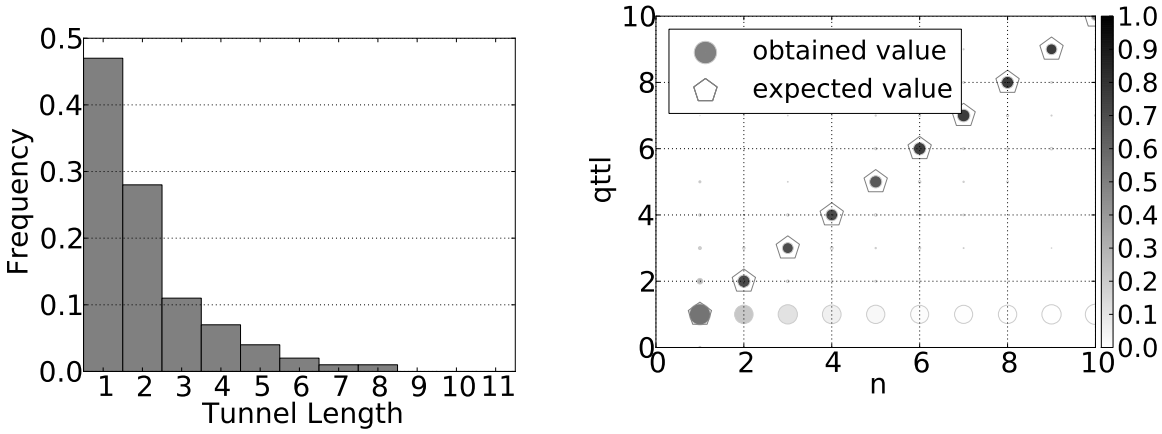
(a) qTTL signature. (b) u-turn signature.

Figura 4.15: Comportamiento de las firmas para los túneles MPLS implícitos

1. **Firma *q-ttl*:**

   Nuestra validación de la firma se basa en hipótesis de que la *posición MPLS* real coincide con la *n-posición*, esto es, la *n-posición* del LSR dentro del LSP corresponde al valor *q-ttl* generado en tal LSR ($q - ttl = n$).

   A fin de validar esta suposición hemos usado el *dataset* obtenido de la exploración detallada previamente y comparado el *q-ttl* con la *n-posición* para túneles explícitos e implícitos. Los resultados son mostrados en la figura 4.16, donde se observa la distribución de la longitud de los túneles MPLS computada como el numero de LSR dentro del túnel. Observamos, corroborando estudios previos, que la mayoría de los túneles son mas bien cortos ($L < 3$ en mas del 80 % de los casos).



(a) Distribución de longitud de túnel (b) Comparación entre *q-ttl* y *n-posición*

Figura 4.16: Comparación entre valores para *q-ttl* y *u-turn* obtenidos y esperados

La figura 4.16(a) también proporciona posibles valores para el *q-ttl*. Esto sugiere así que el valor *q-ttl* debe oscilar entre 1 y 8, con una fuerte predominancia para valores pequeños (esto es, entre 1 y 3).

La figura 4.16(b) representa un *scatter plot* mostrando la relación entre el *q-ttl* y la *n-posición*. El tamaño de los círculos en el *scater plot* esta relacionado con la frecuencia de ocurrencia de la *n-posición* respecto de cada valor en el eje vertical. Por ejemplo, para valores donde $n > 1$, el circulo mas grande está principalmente localizado en *q-ttl = 1* y *q-ttl = n*. Entonces, esto sugiere que, para una dada *n-posición*, el valore *q-ttl* usualmente toma valores *1* o *n*, aunque notamos que coincide mayoritariamente con *n*. El *bias* $q - ttl = n \pm \epsilon$ podría ocurrir debido a dos causas: la primera es la limitación en nuestro método para revelar el primer LSR del LSP cuando la RFC4950 no está implementada (por definición de túnel implícito); la segunda causa podría ocurrir debido a balanceadores de carga (usando `paris-traceroute` deberíamos evitar problemas de balance de carga excepto en los casos que el balanceo sea hecho por paquete) como sugiere la figura 4.14. En la figura 4.16(b) también se muestra que la frecuencia *q-ttl* toma valores *1*, incluso para $n > 1$, lo cual significa que el LSR implementa la RFC4950 pero no coincide con la forma *q-ttl*.

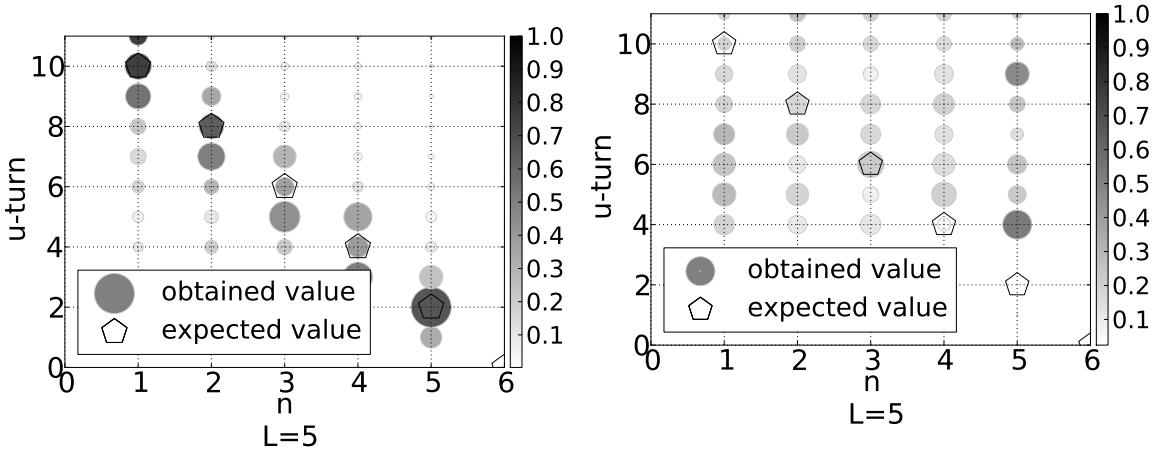
También encontramos que cerca de un 2 % de los LSR no reacciona a la firma *q-ttl*, incluso si sus vecinos sí lo hacen, esto es, algunas interfaces de LSR localizadas en la posición $i_{n\pm1}$) del túnel responden correctamente a la firma *q-ttl* pero la interfaz del LSR localizada en i_n no lo hace.

Sin embargo, la *n-posición* es altamente confiable y por tanto, la potencial presencia de balanceadores de carga los LSP no es un problema común. En efecto, encontramos

que en el 58 % de los casos la *n-posición* coincide con el valor *q-ttl* mientras que en el 36.3 % de los casos la firma *q-ttl* no está presente en túneles explícitos y toma el valor *1* y solo 6.7 % de los casos presentados tiene algún *bias* cerca del valor de n esperado. Tales resultados apoyan nuestra hipótesis: la *posición MPLS* del túnel es altamente coincidente con la *n-posición*. De este modo, nosotros usamos la *n-posición* como un valor de referencia para validar las firmas *u-turn*.

2. **Firma *u-turn*:** como se ha explicado previamente, el valor *u-turn* esperado es de la forma $[2L, 2L-2, 2L-4, ..., 2]$ donde L es la longitud del túnel y el vector de posición corresponde con la posición del LSR dentro del LSP, esto es, n (ver figura 4.15(b)). La relación entre L, n y el valor esperado puede ser escrita como $u-turn = 2\Delta(L-(n-1))$.

   Debido a que la firma *u-turn* esta comúnmente presente en casi todos los LSR, primero, comparamos n con el valor *u-turn* en los LSR hallados explícitamente o los descubiertos mediante la firma *q-ttl* usando el *dataset* generado en nuestra exploración (figura 4.17(a)). También hemos estudiado el valor de n en los LSR donde la firma *u-turn* fue la unica detectada (figura 4.17(b)). Para esto hemos usado el filtro $u - turn > 3$ para evitar falsos positivos (esto es, evitar tuneles pequeños donde los biases son mas probables que aparezcan).



(a) *u-turn* en los LSR revelados a través de la firma *u-turn*

(b) *u-turn* en los LSR revelados donde no está activada la RFC4950 y la firma *q-ttl* no está presente

Figura 4.17: Comparación entre valores obtenidos y esperados para la firma *u-turn*.

Los resultados para una dada longitud de túnel $L = 5$ son mostrados en la figura 4.17 junto a un *scatter plot* que debe leerse de la misma manera que la figura 4.16(b). La figura 4.17 sugiere que la firma *u-turn* es usualmente sobrestimada. Resultados similares fueron observados para otras longitudes de túnel.

Notamos que los valores *u-turn* obtenidos están cerca de los esperados cuando el LSR fue revelado explícitamente o cuando el *q-ttl* está presente (figura 4.17(a)). Sin embargo, para los LSR revelados solamente por la firma *u-turn* (figura 4.17(b)) el valor obtenido y esperado comúnmente no coinciden. Si aceptamos un *bias* de ∓ 2 al rededor del valor *u-turn* sobre nuestro *dataset* completo, notamos que, para los LSR explícitamente revelados o mediante *q-ttl*, el 60 % ode las firmas *u-turn* obtenidas coinciden con el valor esperado. Sin embargo, para los LSR revelados unicamente mediante *u-turn* solo coinciden en menos del 25 % con el valor esperado.

Por lo tanto, los LSR revelados solamente a través de la firma *u-turn* son altamente imprecisos, principalmente, porque los túneles MPLS no son los únicos responsables por las firmas *u-turn*. En efecto, también están relacionados los balanceadores de carga en las rutas de regreso en donde los *ICMP echo-replay* e *ICMP time-exceeded* en diferentes *hops* pueden sufrir balanceo de carga.

Conclusiones

La utilización de la tecnología MPLS tiene un uso extendido en la estructura de Internet. Se encontró que en la exploración realizada cerca del 34 % de los *traceroutes* atraviesan túneles explícitos y 42 % túneles ya sea explícitos o implícitos. Se probó la detección de túneles implícitos, en la que los resultados sugieren que aunque `paris-traceroute` trabaja apropiadamente (evitando balanceadores de carga por paquetes) para descubrimiento de túneles MPLS, las firmas *u-turn* están comúnmente sesgadas debido a que el balanceo por flujo genera problemas en los caminos de regreso.

Relacionado con la presente tesis, hemos mostrado como `Magallanes` puede ser utilizado para la realización de trabajos de investigación importantes otorgando a los investigadores la facilidad de realizar exploraciones amplias sin un gran esfuerzo. Para ejemplificar, en el trabajo [40] se adquirieron datos similares a los utilizados en esta publicación, pero en ese entonces fueron necesarias varias semanas de trabajo para adquirir los datos, mientras que en este último la adquisición de datos se realizó en solo 3 días, sin la necesidad de dedicar mucho de tiempo de trabajo en cada uno de estos.

4.3. Medición del RTT en los primeros hops

El `RTT` (del ingles, *Round-trip delay time*) es el tiempo que tarda un paquete de datos enviado desde un emisor en volver a este mismo habiendo pasado por el receptor de destino. Este medida es importante dado que interviene de modo crucial en la eficiencia de numerosos sistemas como por ejemplo, durante la carga de una página Internet utilizando el protocolo HTTP 1.0, la descarga de cada elemento de la página necesita la apertura y la cierre de una

conexión TCP, por lo que la duración de descarga del elemento es necesariamente superior a 2 RTT.

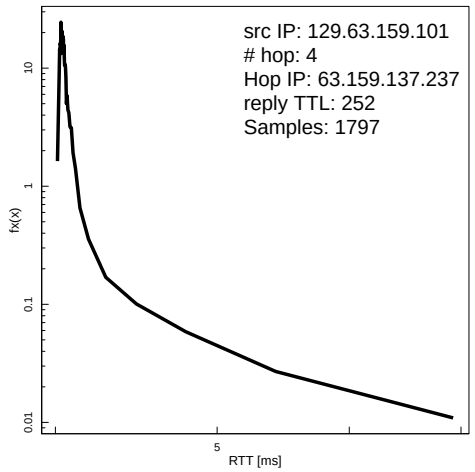
Miembros de CoNexDat han estado estudiando el comportamiento del RTT obtenido en los primeros *hops* en distintas redes utilizando Magallanes. Para esto se han realizado mediciones desde 107 nodos de PlanetLab distribuidos en todo el mundo, con un TTL máximo de valor 4, durante 168 horas (1 semana). Una vez recopiladas las mediciones se debieron encontrar 428 casos, pero debido a variaciones en las rutas y/o balanceo de carga durante la exploración, se llegaron a encontrar 463 casos diferentes. La clasificación utilizada fue basándose en la cuaterna $(ip_{src}, ip_{hop}, hop_N, reply_{TTL})$.

En todos los casos se observó que el comportamiento a lo largo de la semana fue similar, variando levemente los parámetros, pero conservando la misma naturaleza. A su vez, se encontró que en 205 casos se ajusta mejor la *distribución estable* [41] que otras distribuciones clásicas como que la *Weibull*, la *Lognormal* o la *Shifted-Gamma*. Las resultados halladas se han podido agrupar en 4 tipos de respuestas características, las cuales se muestran en la figura 4.18.

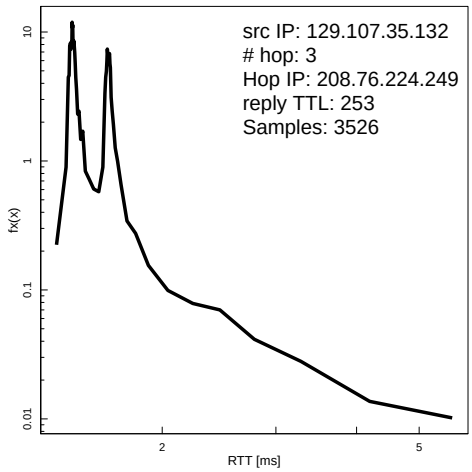
En las respuestas tipo 1 y 3 se observan funciones de densidad empíricas que se pueden caracterizar a través de una *distribución estable* . La diferencia entre las respuestas 1 y 3 es la pendiente o persistencia del fenómeno de decaimiento.

La respuesta tipo 2 muestra un comportamiento bimodal, donde no se detectan múltiples caminos al separar por la cuaterna $(ip_{src}, ip_{hop}, hop_N, reply_{TTL})$. Una hipótesis es que en estos casos existe un balance de carga en el camino de retorno.

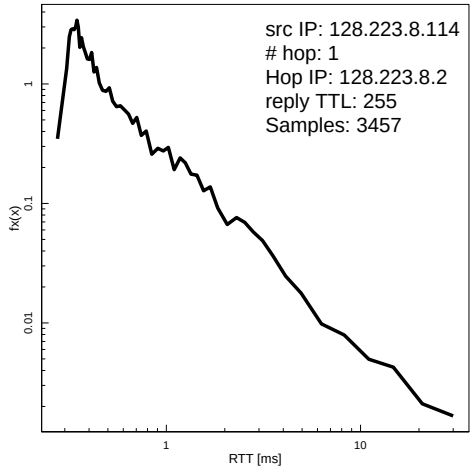
La respuesta tipo 4 presenta un comportamiento que es más difícil de explicar con sólo una variable aleatoria, pero que posiblemente se pueda explicar con una mezcla de distribuciones estables.



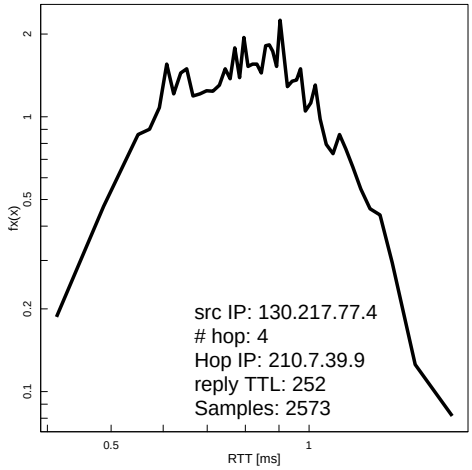
(a) Respuesta tipo 1



(b) Respuesta tipo 2



(c) Respuesta tipo 3



(d) Respuesta tipo 5

Figura 4.18: Distribuciones características del `RTT` hallado en las mediciones

Capítulo 5

Conclusiones

5.1. Contribuciones de esta tesis

Como resultado del trabajo hemos desarrollado un programa, `Magallanes`, para gestión de exploraciones de Internet que cumple con los requisitos que nos propusimos inicialmente. Luego hemos explicado su funcionamiento y los problemas con los que nos hemos topado durante su desarrollo junto a las soluciones implementadas.

A continuación, hemos realizado distintos estudios utilizando esta herramienta. En primer lugar realizamos una exploración de la estructura de Internet, que nos permitió hacer una caracterización a rasgos generales de su topología. Sumado a esto, se utilizó `Magallanes` para llevar a cabo trabajos de investigación. Uno sobre la caracterización del parámetro RRT en los primeros *hops* de distintas redes, y luego otro trabajo sobre la influencias de las redes MPLS dentro de la estructura de Internet, que a su vez desembocó en la publicación [2].

5.2. Trabajos futuros

Para continuar con el desarrollo de este trabajo surgen varias posibilidades de investigación. Por un lado se podría hacer un análisis de las redes por regiones geográficas, es decir, nosotros en el presente trabajo tratamos a la red como un todo, pero podríamos desear caracterizar a la red en cada continente y luego comparar su estructura. A su vez se podría ampliar el análisis de la topología incluyendo peso a los enlaces de modo que representen distintos parámetros como pueden ser latencias, disponibilidad o tase de perdida. Otra posibilidad sería integrar una herramienta para mapear el prefijo IP al ANS de cada nodo y luego, mediante la información provista por el proyecto *route views*, realizar el mapeo de cada nodo al sistema autónomo al que pertenece y crear así mapas a nivel de sistema autónomo.

Apéndice A

Manual de Magallanes

A.1. Breve resumen de Magallanes

`Magallanes` es una herramienta creada para correr exploraciones en Internet utilizando la plataforma PlanetLab, en la cual a través de sus nodos es posible hacer `traceroutes`, `ping` y resolución de aliases desde varios monitores a lo largo del planeta para luego almacenar los resultados en una base de datos central.

El objetivo de `Magallanes` es ayudar a la comunidad científica a conseguir datos de una manera sencilla para sus investigaciones, de modo tal que puedan ahorrar tiempo en esta etapa de sus investigaciones. Lo que `Magallanes` ofrece es:

- Gestión de *slices* de PlanetLab
- Procesos para generar exploraciones mediante *traceroutes* con una amplia variedad de opciones de configuración, entre las que se encuentran:
  - Selección de monitores por ID o en forma aleatorios
  - Selección de *target* por IP o en forma aleatorio, tanto en tamaño como en distribución.
  - Duración de la exploración
  - Periodo de muestreo
  - Distintos tipos de paquetes a utilizar: `icmp-paris`, `udp-paris`, `tcp-paris`, `udp`, `icmp`, `tcp`, `tcp-ack`
  - Tasa de paquetes con las que se se realizan las mediciones
  - Reducción de carga en los *hops* alcanzados por los traceroutes
- Resolución de aliases
- Gestión de la base de datos donde se almacenan las exploraciones

A.2. Instalación

A continuación se explica el procedimiento para la instalación de `Magallanes`. Inicialmente, se requieren los siguientes items:

- *Slice* de PlanetLab
- Sistema operativo derivado de `Debian` en el servidor local
- Python 2.7, con las siguientes bibliotecas:
  - `xmlrpclib`
  - `json`
  - `pexpect`
  - `psycopg2`
  - `netaddr`
  - `geoip2`
- `postgres` 9.4 or superior
- `parallel-ssh`
- `sc_warts2json` (Esta herramienta es instalada junto con `Scamper`)

y descargar la base de datos libre `GeoLite2` de MaxMind:

- `http://geolite.maxmind.com/download/geoip/database/GeoLite2-City.mmdb.gz`
- `http://geolite.maxmind.com/download/geoip/database/GeoLite2-Country-CSV.zip`

De los dos archivos previos, una vez descomprimidos, se deben mover los siguientes archivos a la ruta ≪ */home/.../magallanes/src/files* ≫:

- GeoLite2-Country-Blocks-IPv4.csv
- GeoLite2-Country-Locations-en.csv
- GeoLite2-City.mmdb

Luego se debe configurar una base de datos en `Postgres`. Lo primero es crear la base de datos vacía y configurar los parámetros BD_name, BD_user, BD_host, BD_pass en el archivo de configuración:

≪ *home/.../magallanes/src/files/program.conf* ≫

Luego se continua configurando los restantes parámetros del archivo de configuración:

- ssh_time_out: *timeout* de la conexión *ssh*. Recomendado 45.
- publicKeyRSA: ruta a la llave publica con la que se establece la conexión a los nodos del slice.
- ssh_pass: clave *ssh*
- num_thread: cantidad de hilo con la que se estable conexión a los nodos para tareas paralelizables. Recomendado 15.
- midar_folder: carpeta en la cual se extrae `Midar` luego de ser descomprimido. Por ejemplo: midar-0.5.0.
- num_ip_list: cantidad máxima de direcciones IP para resolver en el proceso de resolución de aliases por nodo. Recomendado 35.000 (Max 40.000)

Una vez hecho esto se debe preparar el archivo *filesToInstall.tar.gz*, el cual contendrá los programas y paquetes que se instalarán en los nodos. Para esto primero se deben descargar los siguientes items:

- `scamper` (http://www.caida.org/tools/measurement/scamper/)
- `midar` (http://www.caida.org/tools/measurement/midar/)
- `mper` (https://www.caida.org/tools/measurement/mper/)
- `rb-perio` (http://www.caida.org/tools/measurement/rb-mperio/)
- `arkutil` (http://www.caida.org/tools/utilities/arkutil/)

y generar el archivo *filesToInstall.tar.gz*, de modo que se deberán tener los siguiente archivos en él:

- 'zlib-devel-1.2.5-2.fc14.i686.rpm'
- 'scamper-cvs-20XXXXXX.tar.gz'
- 'rb-mperio-X.X.X.gem'
- 'mper-X.X.X.tar.gz'
- 'midar-X.X.X.tar.gz'

- 'arkutil-X.X.X.gem'
- 'fedora-updates-testing.repo'
- 'fedora-updates.repo'
- 'fedora.repo'

El siguiente paso es revisar en ≪ */home/.../magallanes/src/installNodes.sh* ≫ si las *variables* son correctas respecto de las versiones de los programas que se han agregado al *filesToInstall.tar.gz*.

Opcionalmente, se puede editar el archivo ≪ */home/user/.bashrc* ≫ agregando la siguiente linea: `alias magallanes='python /home/.../magallanes/src/main.py'`, de tal modo que se pueda ejecutar el programa simplemente tipeando `'magallanes'`.

A.3. Utilización

Al iniciar `Magallanes`, la primer pantalla es el *Menú de usuario* en el cual se tienen tres opciones:

1. **Users**: acceder con un usuario ya registrado.
2. **Manual**: acceder ingresando el usuarios y contraseña de PlanetLab.
3. **Options**: agregar/quitar usuarios.

Una vez que se ingresa mediante la opción *users*, aparecerán los *slices* disponibles en su cuenta. Se debe elegir con cual trabajar, y a continuación `Magallanes` comprobará la conexión *ssh* con cada uno de los nodos asociados al mismo. Terminada esta comprobación se aparecerá el *Menú principal*, desde el cual se accede a las dos secciones principales: 1. **Administración de nodos**; 2. **Administración de exploraciones**.

1. **Admin nodes**
   - I View nodes: muestra información de todos los nodos asociados al slice en uso.
   - II Add nodes: agregar nodos al slice. Tener en cuenta que existe un retardo de un día aproximadamente entre el momento que se agrega un nodo al slice y que pasa a ser utilizable.
   - III Install software in nodes: instalación en los nodos del software requerido para realizar las exploraciones.
   - IV Remove nodes: quitar nodos del slice.

V Update information of Planetlab nodes: actualiza la tabla en la cual se cuanta con la información de todos los nodos de PlanetLab

VI Update table of ipv4 address blocks: actualiza la tabla donde se guardan los bloques de direcciones IP obtenidos de MaxMind.

2. **Admin explorations**

I View functional nodes: muestra los nodos 'utilizables', esto es, los nodos con conexión ssh y el software requerido instalado.

II Summery of last explorations: breve resumen de las ultimas exploraciones realizadas.

III New exploration: preparación y ejecución de una nueva exploración.

IV Cancel exploration: cancela una exploración no almacenada, esto es, detiene los procesos y elimina los archivos residuales en los nodos.

V Stored exploration: descarga los datos generado de una exploración y los almacena en la base de datos local.

VI Alias resolution: preparación y ejecución del proceso de resolución de aliases.

VII Delete data: elimina la información referente a una exploración de la base de datos.

Cuando se prepara una exploración (opción *new exploration*) se tienen los siguientes parámetros a establecer:

1. Nodes: selección de los nodos origen (monitores) desde donde se ejecutarán los *traceroutes*.

2. Kind of target: desde donde y con que criterio se elegirá el target.

   Para correr una exploración exhaustiva, la opción apropiada es *3. Internet; Random* debido a que esta permite seleccionar muchas direcciones IP distribuidas a lo largo del planeta. El criterio de elección de estas IP consiste en tomar una IP aleatoria de cada bloque de direcciones obtenido de la base de datos provista por MaxMind. Una vez que una IP es seleccionada de un bloque, tal bloque no será vuelto a utilizar hasta no acabarse la totalidad de bloques disponibles.

3. Period of traceroutes: establece el tiempo mínimo entre las rondas de `traceroutes`, es decir, el tiempo entre que comienza una ronda de `traceroutes` y la siguiente.

4. Duration of exploration: establece la duración de la exploración.

5. Probe type: establece el tipo de `traceroutes` a usar. Los tipos disponibles son: UDP, ICMP, UDP-paris, ICMP-paris, TCP, and TCP-ACK.

6. PPS: establece la tasa máxima de paquetes por segundo que utilizará `Scamper` al momento de ejecutar los `traceroutes`.

7. GATLIMIT: establece la cantidad de `hops` sin respuesta permitidos hasta comprobar si el nodo destino responde.

8. WAIT: establece el tiempo de espera a la respuesta de un `hop`.

9. Recalculatino algorithm: establecer la opción (y el tiempo) de calculo de primer hop.

10. Series of ping: establece si al final de la exploración se realiza una ronda de `ping` a todas las IP halladas en la exploración desde el nodo.

11. Description: comentario opcional acerca de la exploración.

A.4. Base de datos

Resultados de la exploración

Los datos generados en cada exploración son almacenados en la base de datos indicada en el archivo de configuración. Las tablas que contienen los resultados de cada exploración siguen el siguiente esquema:

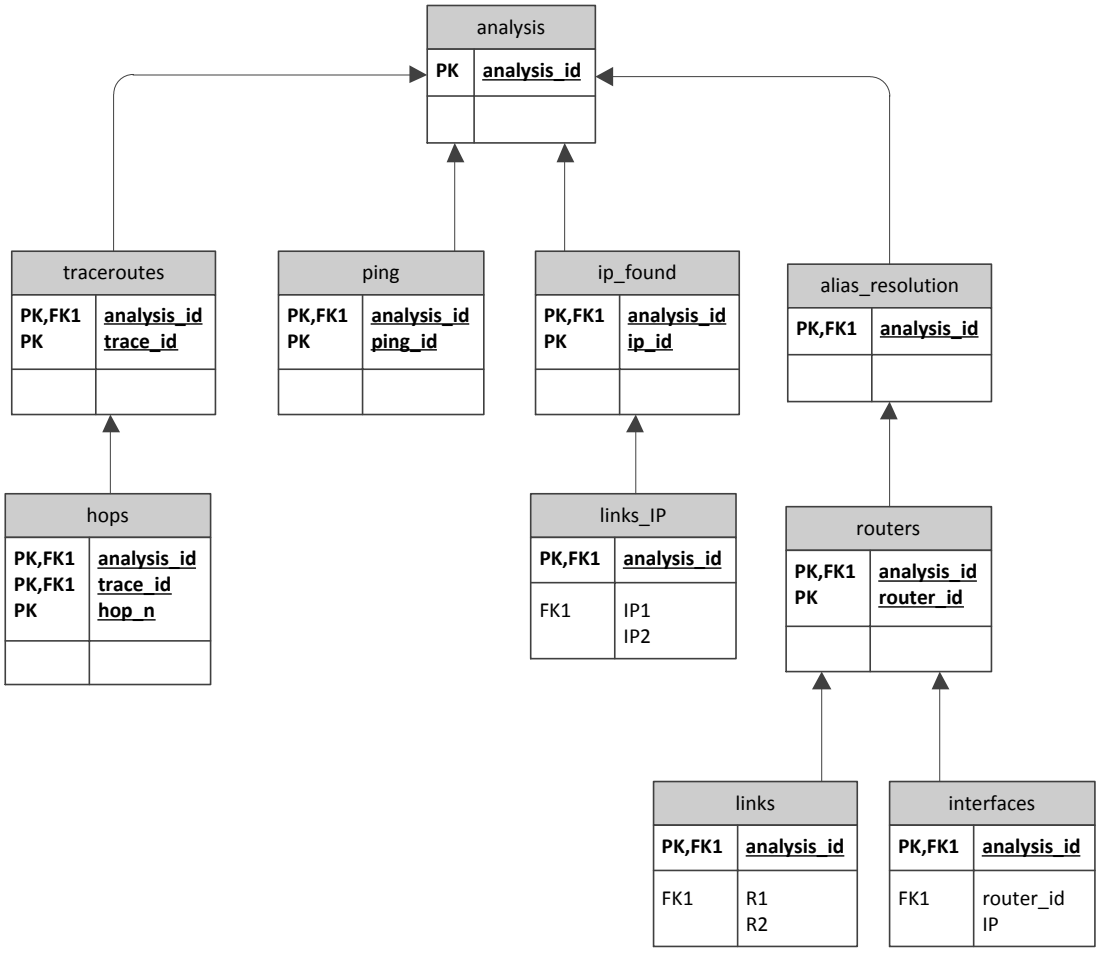


Figura A.1: Esquema simplificado de la base de datos implementada

y la información almacenada en cada una de las tablas es la siguiente:

- ANALYSIS: tabla principal, en la que se encuentra la información general de las exploración.
- TRACEROUTES: cabecera de los `traceroutes` ejecutados en cada exploración.
- HOPS: resultado de los `traceroutes`.
- PING: resultados de los `ping`.
- IP_FOUND: direcciones IP halladas en una exploración.
- LINKS_IP: enlaces a nivel de IP hallados en una exploración.
- ALIAS_RESOLUTION: información acerca del proceso de resolución de aliases.
- ROUTERS: *routers* descubiertos en el proceso de resolución de aliases

- LINKS: enlaces a nivel de *router* obtenidos del proceso de resolución de aliases.
- INTERFACES: interfaces de cada *router* obtenidas del proceso de resolución de aliases.

Los datos presentes de las exploraciones no poseen ninguna clase de procesamiento, lo cual significa que son los datos tal cual los hemos descargados de los nodos. El objetivo de esto es que los investigadores puedan tener un conjunto de datos limpios del cual comenzar sus investigaciones.

Administración interna

Además de tablas donde se almacenan las exploraciones, `Magallanes` posee las siguientes tablas de uso interno:

- USERS: usuario y contraseña de los usuarios con la que acceden a PlanetLab
- NODES_WORKING: lista de nodos en los que se está corriendo una exploración en cada momento
- IP_RESOLUTION: para la gestión del proceso de resolución de aliases. Almacena la lista provisoria de IPs resuelta durante el proceso.
- IP_METODO_MIDAR: para la gestión del proceso de resolución de aliases. Almacena el *metodo preferido* obtenido en la fase de estimación.
- PLANETLAB_NODES: información de los nodos de PlanetLab
- ADDRESS_BLOCK_IPV4: bloques de direcciones obtenidos de MaxMind
- BLACKLIST_NODES_EXPLORATION: lista de nodos en los que no se ejecutan exploraciones
- NODES_PRIORITY: prioridad en la elección de nodos para realizar el proceso de resolución de aliases. A igual prioridad, la elección es aleatoria.

Bibliografía

[1] *Magallanes*. https://github.com/ihameli/magallanes.

[2] Fernando Davila Revelo, Mauricio Anderson Ricci, Benoit Donnet y José Ignacio Alvarez-Hamelin. «Unveiling the MPLS Structure on Internet Topology». En: *8th International Workshop on Traffic Monitoring, Analysis (TMA)* (2016). URL: http://orbi.ulg.ac.be/bitstream/2268/194694/1/paper.pdf.

[3] A. Tanembaum. «Internet». En: *Redes de computadoras, 4ta Edicion*. Pearson Educacion, 2003. Cap. 1.5.1, págs. 50-59.

[4] *Breve historia de internet*. http://www.internetsociety.org/es/breve-historia-de-internet.

[5] *World Internet Users, 2015 Population Stats*. http://www.internetworldstats.com/stats.htm.

[6] *Internet World Stats*. http://www.internetworldstats.com/stats.htm.

[7] A. Tanembaum. «Redes de computadora». En: *Redes de computadoras, 4ta Edicion*. Pearson Educacion, 2003, pág. 3.

[8] K. Keys. «Internet-Scale IP Alias Resolution Techniques». En: *ACM SIGCOMM Computer Communication Review (CCR)* 40.1 (2010), págs. 50-55.

[9] *MIDAR*. http://www.caida.org/tools/measurement/midar/.

[10] *CAIDA:Center for Applied Internet Data Analysis*. http://www.caida.org/.

[11] K. Claffy, T. Monk y D. McRobb. «Internet Tomography». En: *Nature* (1997). URL: http://www.nature.com/nature/webmatters/tomog/tomog.html.

[12] *Archipelago (Ark) Measurement Infrastructure*. http://www.caida.org/projects/ark/.

[13] *Paris Traceroute*. http://www.paris-traceroute.net/.

[14] *CAIDA Research*. http://www.caida.org/research/.

[15] Eran Shir Yuval Shavitt. «DIMES: Let the Internet Measure Itself». En: *ACM SIGCOMM Computer Communication Review* (2005).

[16] *University of Oregon Route Views Project*. http://www.routeviews.org/.

[17] *Internet Topology Datasets Collected on the Archipelago (Ark) Infrastructure*. http://www.caida.org/projects/ark/topo_datasets.xml.

[18] *PlanetLab: An open platform for developing, deploying, accessing planetary-scale services*. https://www.planet-lab.org/.

[19] Larry Peterson y Timothy Roscoe. «The Design Principles of PlanetLab». En: *Operating Systems Review* (2004).

[20] Brice Augustin, Xavier Cuvellier, Benjamin Orgogozo, Fabien Viger, Timur Friedman, Matthieu Latapy, Clémence Magnien y Renata Teixeira. «Avoiding Traceroute Anomalies with Paris Traceroute». En: IMC '06 (2006), págs. 153-158. URL: http://doi.acm.org/10.1145/1177080.1177100.

[21] *Paris Traceroute*. https://paris-traceroute.net/about.

[22] D. Thales y C. Hopps. «Multipath Issues in Unicast, Multicast Next-Hop Selection. RFC 2991.» En: *IETF* (2001). URL: https://tools.ietf.org/html/rfc2991.

[23] *Que es GitHub?* http://conociendogithub.readthedocs.org/en/latest/data/introduccion/.

[24] K. Keys, Y. Hyun, M. Luckie y k. Claffy. «Internet-Scale IPv4 Alias Resolution with MIDAR». En: *IEEE/ACM Transactions on Networking* 21.2 (2013), págs. 383-399.

[25] *MaxMind*. https://www.maxmind.com/es/home.

[26] *Geolocation software*. https://en.wikipedia.org/wiki/Geolocation_software.

[27] R. Albert, H. Jeong y A. Barabasi. «Diameter of the world-wide web». En: *Nature* (1999).

[28] Aaron Clauset, Cosma R. Shalizi y Mark E.J. Newman. «Power-law distributions in empirical data». En: *SIAM* 4.51 (2009), págs. 661-703. URL: http://arxiv.org/pdf/0706.1062.pdf.

[29] M. E. J. Newman. «Assortative Mixing in Networks». En: *Phys. Rev. Lett.* 89 (20 2002), pág. 208701. DOI: 10.1103/PhysRevLett.89.208701. URL: http://link.aps.org/doi/10.1103/PhysRevLett.89.208701.

[30] Jose Ignacio Alvarez-Hamelin. «Taxonomia de los Modelos de Topologia de Internet». En: *Mecanica Computacional, Interdisciplinary Mathematical Methods*. Vol. XXV. 29. CIMEC, 2006, págs. 2597-2612. URL: http://www.cimec.org.ar/ojs/index.php/mc/article/view/636/604.

[31] Sun Jimeng y Tang Jie. «A survey of models, algorithms for social influence analysis». En: *Social network data analytics* (2011).

[32] Mariano G. Beiró, J. Ignacio Alvarez-Hamelin y Jorge R. Busch. «A low complexity visualization tool that helps to perform complex systems analysis». En: *New J. Phys* 10.12 (2008), pág. 125003. URL: http://dx.doi.org/10.1088/1367-2630/10/12/125003.

[33] J. Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat y Alessandro Vespignani. «Large scale networks fingerprinting and visualization using the k-core decomposition». En: *Advances in Neural Information Processing Systems 18*. Ed. por Y. Weiss, B. Schölkopf y J. Platt. Cambridge, MA: MIT Press, 2006, págs. 41-50. URL: http://cnet.fi.uba.ar/ignacio.alvarez-hamelin/pdf/AH_D_B_V_NIPS2006.pdf.

[34] E. Rosen, A. Viswanathan y R. Callon. «Multiprotocol Label Switching Architecture, RFC 3031». En: *IETF* (2000). URL: http://www.rfc-editor.org/rfc/rfc3031.txt.

[35] R. Bonica, D. Gan, D. Tappan y C. Pignataro. «ICMP Extensions for Multiprotocol Label Switching. RFC 4950.» En: (2007). URL: http://www.rfc-editor.org/rfc/rfc4950.txt.

[36] Joel Sommers, Paul Barford y Brian Eriksson. «On the Prevalence, Characteristics of MPLS Deployments in the Open Internet». En: *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference* (2011). URL: http://doi.acm.org/10.1145/2068816.2068858.

[37] B. Donnet, M. Luckie, P. Merindol y J. Pansiot. «Revealing MPLS tunnels obscured from traceroute». En: *ACM SIGCOMM Computer Communication Review (CCR)* 42.2 (2012), págs. 87-93.

[38] J. Postel. «Internet Control Message Protocol. RFC 3812.» En: *IETF* (1981). URL: https://tools.ietf.org/html/rfc3812.

[39] Yves Vanaubel, Jean-Jacques Pansiot, Pascal Merindol y Benoit Donnet. «Network Fingerprinting: TTL-Based Router Signature». En: *CM/USENIX Internet Measurement Conference (IMC)* (2013).

[40] Fernando Davila. *Impacto de los túneles MPLS en la topología de Internet*. 2014. URL: http://cnet.fi.uba.ar/fernando_davila/Tesis_Fernando_Davila.pdf.

[41] Vladimir V. Uchaikin y Vladimir M. Zolotarev. «Chance and Stability. Stable Distribution and their Applications». En: (1999).