Universidad ORT Uruguay
Facultad de Ingeniería

Contribución Proyecto Simón

Ampliación y optimización de mediciones de latencia para la región de América Latina y el Caribe

Entregado como requisito para la obtención del título de Ingeniero en Telecomunicaciones

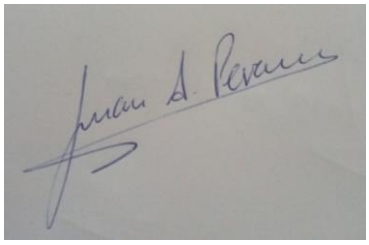
Juan Alejo Peirano - 162129
Tutor: Nicolás Antoniello

2015

Declaración de autoría

Yo, Juan Alejo Peirano, declaro que el trabajo que se presenta en esa obra es de mi propia mano. Puedo asegurar que:

- La obra fue producida en su totalidad mientras realizaba el proyecto de fin de carrera;
- Cuando he consultado el trabajo publicado por otros, lo he atribuido con claridad;
- Cuando he citado obras de otros, he indicado las fuentes. Con excepción de estas citas, la obra es enteramente mía;
- En la obra, he acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, he explicado claramente qué fue contribuido por otros, y qué fue contribuido por mí;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.



Juan Alejo Peirano
28/10/2015

Agradecimientos

En esta instancia me gustaría agradecer el apoyo del *staff* de LACNIC, en particular de Agustín Formoso, por su disposición para realizar reuniones en todo momento y fomentar discusiones importantes durante todo el proyecto.
También me gustaría agradecer a Lucía Silva, por facilitar un excelente ambiente de trabajo durante la realización del proyecto.

Abstract

El proyecto Simón, iniciativa de LACNIC, Registro de direcciones de Internet para América Latina y el Caribe, tiene como objetivo principal brindar mediciones recientes y de calidad acerca de la conectividad de Internet en la región de Latinoamérica y Caribe. Esta plataforma actúa como centro de recopilación de estas mediciones, brindando los datos a los principales agentes de la región como pueden ser proveedores de contenido, encargados de infraestructura o proveedores de servicios.

El proyecto Simón comenzó en 2009 como una iniciativa por parte de aficionados a Internet, y a lo largo del tiempo se ha ido renovando con nuevas tecnologías y nuevas funcionalidades. Con esta contribución, se buscan renovaciones, tanto técnicas como conceptuales, que mejoren sustancialmente la calidad de los datos recabados, la performance del medidor principal y una ampliación de los resultados obtenidos hoy en día.

Mediante el desarrollo y modificación de secciones del proyecto, se obtuvieron:

- Resultados de comportamiento de actores involucrados en las mediciones de latencia, realizando un total de 25 pruebas, de 1000 mediciones de latencia cada una, compuestas en una matriz de 4 Sistemas operativos y 5 navegadores web.
- Comparaciones de 4 medidores con características diferentes.
- Más de 20.000 mediciones (en ambientes de maqueta) de 2 organizaciones internacionales que aportan distintas técnicas y metodologías a las contenidas en el proyecto.
- Acceso a nueva información de infraestructura regional, utilizando un Punto de Intercambio de Trafico como referencia para, a partir de ahí, obtener información del resto de los Puntos en el futuro.
- Una matriz de latencias y ASNs basada en más de 160.000 mediciones de latencia.
- Capacidad de obtener medidas hacia distintos destinos establecidos, de 2 maneras adicionales a la contenida inicialmente en el proyecto
- Un pre-análisis de un indicador de conectividad en la región de influencia de LACNIC

Se concluye por lo tanto, que la contribución realizada ha otorgado un beneficio al proyecto, en la ampliación y optimización de mediciones de latencia y a su vez se han fomentado actividades para continuar desarrollando en el futuro.

Palabras Clave

LACNIC, Simón, latencia, ASN, IP, *ping, traceroute.*

Índice

Introducción 8
Actividades proyectadas 11
Capítulo 1 - Familiarización con las herramientas de trabajo y el entorno del proyecto 13
1.1 Introducción 13
1.2 Lenguaje de programación Python 13
1.3 Lenguaje de programación JavaScript 13
1.4 Framework Django 13
1.5 Objetivo, arquitectura y funcionamiento de Simón 14
1.6 Entorno de trabajo de Simón 22
Capítulo 2 - Normalización de datos 25
2.1 Normalización de los resultados 25
2.1.1 Introducción 25
2.1.2 Medidor TCP 27
2.1.3 Medidor NTP 29
2.1.4 Medidor JavaScript 31
2.1.5 Medidor ICMP 33
2.1.6 Comparaciones 34
2.2 Normalización de los navegadores y sistemas operativos 36
2.2.1 Introducción 36
2.2.2 Definición arquitectura de maqueta 37
2.2.3 Configuración y programación de maqueta 45
2.2.4 Resultados 55
Capítulo 3 - Incorporación de mediciones de partners internacionales 58
3.1 Introducción general 58
3.2 RIPE NCC 59
3.3 Obtención de datos de RIPE Atlas 64
3.3.1 Introducción 64
3.3.2 Etapas definidas para obtención de datos 66
3.4 CAIDA 74
3.5 Obtención de datos Archipielago (Ark) - CAIDA 77
3.5.1 Introducción 77
3.5.2 Etapas definidas para obtención de datos 82
Capítulo 4 - Incorporación de Sistemas Autónomos 85

4.1 Introducción 85
4.2 Matriz de latencias por AS 86
4.2.1 Comentarios 87
4.3 Relación largo de AS-Path y latencia 89
Capítulo 5 - Incorporación de Puntos de Intercambio de Tráfico 92
5.1 Introducción 92
5.2 Comparación de medidas 93
Capítulo 6 - Modificaciones en Plataformas actuales 96
6.1 Introducción 96
6.2 Modificaciones realizadas 97
Capítulo 7 - Índice de conectividad 99
Capítulo 8 - Resultados 101
Capítulo 9 – Conclusiones 111
Referencias 115
ANEXO 122

Introducción

El proyecto Simón [1] busca crear herramientas que aporten valor a la hora de tomar decisiones en lo que respecta a la planificación, operación y análisis de Internet en la región de Latinoamérica y el Caribe creando un repositorio único, abierto y confiable que contenga el estado de algunas de las variables y mediciones más importantes que tiene la red.
Dentro de las variables mencionadas, se encuentra la latencia, uno de los factores más descriptivos a la hora de evaluar velocidad (*throughput*) y calidad de las conexiones a Internet de los usuarios.
La medición de latencia ha sido el objetivo principal del proyecto Simón en los últimos años, aplicando diferentes técnicas que contemplen las tecnologías y aplicaciones utilizadas actualmente [2]. De esta manera se busca a través de la medición de latencias y estudio de la interconexión, generar información que permita a las empresas y organizaciones de la región de LACNIC, realizar casos de negocio que permitan el desarrollo y la interconexión a nivel local y regional. [3]
Se pueden identificar a los siguientes actores y/o público objetivo (entre otros) como usuarios de los datos y las plataformas desarrolladas por Simón:

- Proveedores de Servicios de Internet tanto fijos como móviles a usuarios finales.
- Proveedores de tránsito de datos en la región.
- Organizaciones sin fines de lucro como Universidades y ONGs que desarrollen actividades vinculadas a Internet en la región
- Organizaciones y empresas relacionadas a la generación de estadísticas.
- Entidades gubernamentales.
- Usuarios particulares que deseen conocer y/o investigar sobre el estado y conectividad en la región

Al día de hoy se mide latencia en la región, principalmente basándose en algoritmos que consultan de manera pseudo-aleatoria servidores base en Internet utilizando como origen aplicaciones (en general WEB) en el propio equipo de quién realiza la medición. En particular, consultas a servidores de mediciones de velocidad de conexión (uno de los más utilizados es, el *Speedtest de Ookla*).

Al comienzo de este proyecto, Simón utiliza para la realización de las medidas, el mecanismo mencionado, básicamente corriendo un cliente web en el equipo de quién ejecuta la medida, contra determinados servidores ubicados en diversos puntos de la región (mecanismo que se detalla más adelante en la documentación).

Esta contribución introduce nuevos mecanismos para la incorporación de medidas e información de latencia, agregando datos (y sistematizando dicha agregación) procedentes de otras fuentes como ser en particular los sistemas Atlas de RIPE y Archipiélago de CAIDA, buscando mejorar tanto la calidad de los datos, como las posibilidades de análisis y elaboración de conclusiones a partir de los mismos.

Si bien más adelante se detalla el funcionamiento tanto de Atlas como de Archipiélago, se deben tener en cuenta algunos aspectos puntuales sobre cada uno de los dos sistemas mencionados que permitan visualizar desde ya la globalidad del proyecto planteado:

- En el caso del proyecto Atlas de RIPE, las medidas se toman entre dispositivos denominados "*probes*" que se encuentran diseminados en forma pseudo-uniforme en la región (ya que el alojar una de estas *probes* es voluntario de cualquier persona u organización). Además se debe tener en cuenta que Atlas toma datos Globales y no solo de la región, lo que hará necesario "filtrar" de éstos datos globales para quedarse con lo que sea intra-región.
- En el caso de Archipiélago, las medidas se toman desde servidores ubicados en diversos puntos de la región elegidos exclusivamente por CAIDA. Al igual que el caso anterior, será necesario quedarnos únicamente con los datos relevantes de la región.

A partir de lo anterior, se puede inferir que los datos obtenidos del proyecto Atlas representan un espacio muestral bastante grande y distribuido entre puntos de diversas características respecto a los servicios de conectividad con Internet, cada uno arrojando un número reducido de medidas. Por otro lado los de Archipiélago refieren a un número mucho menor de medidores, no tan distribuido respecto al servicio de conectividad con Internet que lo aloja, pero arrojando cada uno una cantidad muchísimo mayor de medidas en un determinado intervalo de tiempo.

La introducción a Simón de estos dos mecanismos de medición, sumado a la actual toma de medidas, representa un muy buen complemento y mejora respecto de la cantidad, calidad y diversidad de datos utilizados para el análisis. Resulta provechoso además, la ventaja de agregarlos al sistema Simón (en contraposición a un eventual manejo de los datos sobre las plataformas individuales), debido entre otros a la regionalización y la posibilidad de cruzamiento y análisis combinado de los mismos.

En consecuencia, mediante las modificaciones realizadas durante este proyecto se pretende:

- Aumentar la base de datos de mediciones actuales, con la colaboración de *partners* internacionales (en particular los mencionados RIPE y CAIDA) y realizar una normalización de estos datos a los ya recabados por el proyecto
- Obtener mediciones entre Sistemas Autónomos (ASNs) y no solamente entre clientes y servidores base.
- Estudio de los caminos que existen entre ASNs, en pos de medir calidades de enlaces.
- Realizar comparaciones y medir el impacto de los Puntos de intercambio de tráfico en la región (IXPs)

- Mejorar las herramientas que existen actualmente en el proyecto.

Actividades proyectadas

A continuación se detallan las actividades y los entregables proyectados como aporte al proyecto Simón. La lista a continuación es simplemente un esbozo general de las actividades que se realizaron durante el proyecto. Varios de los términos utilizados en este resumen se explicarán en detalle en los sucesivos capítulos:

- **Familiarización con las herramientas de trabajo y el entorno del proyecto**
    - Comprensión del proyecto en su estado actual y herramientas utilizadas
- **Normalización de datos**
    - Estudio y tabla de normalización de navegadores y sistemas operativos.
    - Estudio y normalización de resultados.

- **Incorporación de mediciones de *partners* internacionales**
    - RIPE NCC
    - CAIDA

- **Incorporación de Sistemas Autónomos**
    - Matriz de latencias por ASN
    - Estudio de la relación de los *ASN-path* y la latencia

- **Incorporación de Puntos de Intercambio de Tráfico y gestión de mediciones**
    - Modificaciones al código existente con nuevas funcionalidades

- **Índice de conectividad**
    - Estudio sobre viabilidad de Índice de conectividad para la región de América Latina y el Caribe.

- **Mejoras sobre el medidor *JavaScript***
    - Implementación de mejoras al medidor

Durante el proyecto se utilizarán distintas siglas, ya sea en ambientes de programación como de telecomunicaciones. A continuación se detallan algunas de las más utilizadas:

REST - *Representation State Tranfers* [4]
Estilo de arquitectura de software compuesta por guías y mejores prácticas para desarrollar servicios Web escalables.

API – *Application programming interface* [5]
Es un conjunto de componentes, protocolos, rutinas, herramientas, etc. Utilizadas para construir o manipular aplicaciones de software. Una API define funcionalidades independientes de sus implementaciones, lo que permite mediante bloques de programados, crear distintas funcionalidades.

JSON - *JavaScript Object Notation* [6]
Es un estándar abierto que utiliza texto, legible para un ser humano, para enviar objetos de datos que contienen pares del tipo atributo-dato. Es utilizado principalmente para transmitir información entre aplicaciones web y servidores.
Ejemplo de formato JSON:

{"seccion1": {"atributo0": dato0}, "seccion2": [{“atributo1”:dato1, “atributo2”:dato2,…},{“atributo1”:dato1, “atributo2”:dato2,…}]}

El archivo JSON está conformado por 2 secciones, la primera con un solo par atributo-dato, la segunda sección con una lista de conjuntos de pares atributo-dato.

RTT – *Round Trip Time* [7]
Es la cantidad de tiempo (en general medida en microsegundos) que demora una señal en llegar a un receptor en particular, más la cantidad de tiempo que demora en llegar la contestación de esa señal al emisor.

Capítulo 1 - Familiarización con las herramientas de trabajo y el entorno del proyecto

1.1 Introducción

Durante el trascurso de este capítulo se busca introducir el funcionamiento general de Simón y las herramientas utilizadas durante la contribución para el correcto desarrollo de las actividades.

1.2 Lenguaje de programación Python

Python es un lenguaje de programación orientado a objetos. En general es utilizado en el ambiente de las telecomunicaciones por su flexibilidad y sintaxis clara.
En el caso del proyecto será utilizado como lenguaje principal de programación para realizar scripting y modificaciones, ya que el proyecto Simón se basa en el Framework *Django,* basado en *Python* (además de otros lenguajes) que se explican a continuación.
Durante el transcurso de este proyecto utilizaremos la versión 2.7 de *Python* [8].

1.3 Lenguaje de programación JavaScript

JavaScript es un lenguaje de programación ágil orientado a objetos, desarrollado principalmente para ser utilizado en plataformas web. A pesar de compartir parte del nombre, JavaScript no depende de Java, es una plataforma de programación distinta, con similitudes en su sintaxis con el lenguaje de programación "C". A lo largo del proyecto, *JavaScript* será utilizado de la misma manera que *Python*, pero apuntando a la programación de scripts a aplicaciones web.

1.4 Framework Django

El entorno de programación a utilizar es *Django* [9]. La razón de elegir este *framework* en particular, es primeramente por ser el utilizado en el proyecto Simón, que incorpora herramientas de desarrollo web en conjunto con características de programación ágil. Desde hace ya algunos años, Simón utiliza como base de mediciones scripts embebidos en páginas web, por lo que *Django* es una herramienta de gran utilidad.

Como características interesantes, se puede resaltar las facilidades que provee el entorno para realizar mapeos de entre objetos de programación y tablas o entradas

en bases de datos (*Django* utiliza por defecto DB *Postgres*) y la capacidad de crear scripts como comandos directos del *framework*. Estas características brindan flexibilidad a la hora de integrar nuevos módulos al desarrollo central.
A su vez, dado que una de las necesidades básicas de Simón es recolectar datos y almacenarlos, tener acceso simple a la base de datos utilizada, es de gran ayuda para programador. En particular nos centraremos en las funciones establecidas en la API *models* [10] [11], la cual contiene los métodos necesarios para configurar y almacenar los resultados en la base de datos que se desee.
A pesar de estas bondades, durante el desarrollo del proyecto se codifican scripts con características más definidas de acceso a base de datos, ya que este tipo de programación permite realizar consultas de manera más rápida a la base de datos, utilizando menos recursos en los ambientes desarrollados. Esto último resulta de gran importancia, como se detallará posteriormente, ya que se utilizan máquinas virtuales y entornos con recursos restringidos para realizar pruebas sobre bases de datos y archivos de texto sumamente extensos.
Durante este proyecto utilizaremos la versión 1.6 de *Django.*

1.5 Objetivo, arquitectura y funcionamiento de Simón

Simón es una herramienta desarrollada por LACNIC y colaboradores, con el objetivo de realizar y almacenar mediciones de latencia en la región de Latinoamérica y el Caribe. Para cumplir con este objetivo, Simón posee diferentes tipos de medidores especialmente diseñados para medir latencias. A continuación se mencionan los medidores que serán explicados en profundidad en los capítulos subsiguientes:

- Medidor TCP: mide el tiempo de respuesta entre el envío de un SYN y la recepción mediante un ACK
- Medidor NTP: mide el tiempo de respuesta de una consulta NTP aleatoria
- Medidor *JavaScript*: mide el tiempo de respuesta de una consulta GET HTTP a una url aleatoria.

Los medidores mencionados anteriormente, son los contenidos en el proyecto Simón, previo a los aportes realizados en esta contribución, ya que con la misma, se agregan las medidas de partners internacionales mencionados en la Introducción.

En esta sección se hará hincapié en el funcionamiento del medidor *JavaScript,* ya que es el más ampliamente utilizado al día de hoy.
Como fue mencionado anteriormente, el proyecto Simón se encuentra desarrollado bajo el *framework Django*, siendo una aplicación dentro del mismo. A su vez *Django* permite realizar desarrollos de páginas web, en base a *templates*.

Por lo tanto se puede definir la arquitectura de Simón en tres componentes básicos:

- **Un FrontEnd web**: donde podemos encontrar visualizaciones, disparar el accionar de ciertos medidores y realizar muestras de datos.
- **La aplicación Simón**: donde se aloja la lógica en referencia a los medidores y comportamiento del FrontEnd web
- **Una base de datos**: donde se almacenan todos los datos

Estos tres elementos conviven dentro de un servidor con sistema operativo *Ubuntu*. Desde el punto de vista gráfico se puede ilustrar como en la Figura 1.1

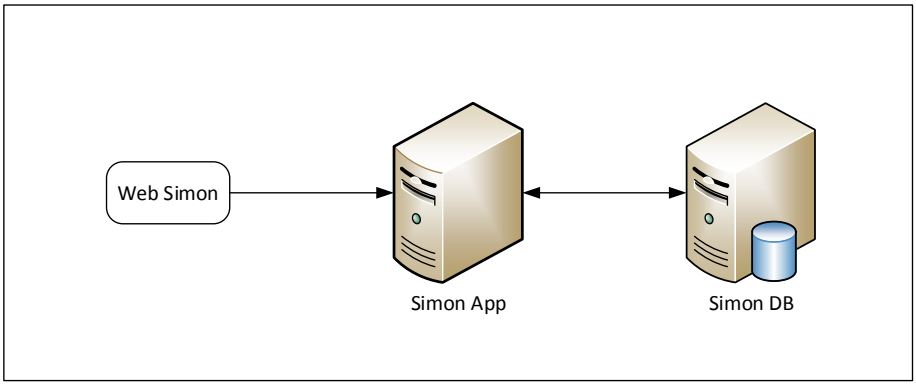


Figura 1.1 – Elementos Arquitectura Simón

Teniendo en cuenta lo elementos que componen a Simón, el flujo de trabajo de realización y almacenamiento de medidas se separar en dos situaciones diferente:

1. Los usuarios navegan en l página del proyecto http://simon.lacnic.net .
2. Los usuarios navegan en una página web que forma parte de los colaboradores del proyecto, distinta a la mencionada en el punto 1.

Ambos casos tienen un funcionamiento similar, pero para el caso 1, la página web está alojada en el mismo servidor que la aplicación, no así para el caso 2, lo que implica tener otras consideraciones.
Por motivos de utilización, que se detallan en el capítulo 2, sección 2.1, en esta sección se tendrá en cuenta solamente el flujo de trabajo del medidor llamado *JavaScript*, siendo el más utilizado por el proyecto en la actualidad.

Un diagrama general correspondiente al caso 1 es el siguiente:

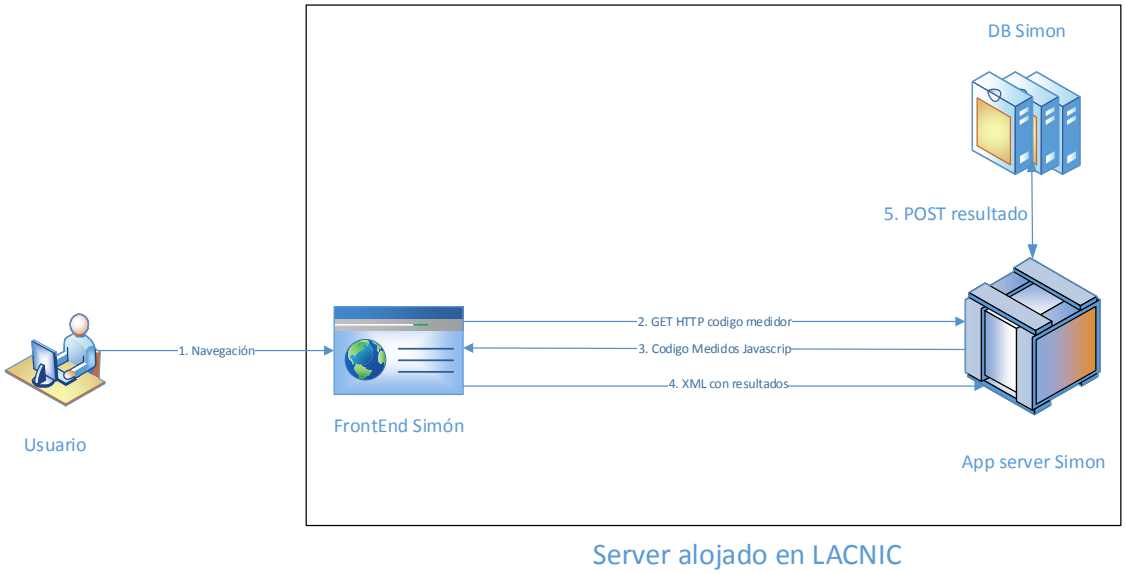


Figura 1.2 – Flujo de obtención de medidas en web Simón

El flujo se representa con los siguientes pasos:

1. Un usuario navega en la web de Simón
2. Dentro de la página web, se solicita el código del medidor para ser ejecutado
3. El servidor de aplicaciones devuelve el código para ser ejecutado de manera transparente en la página web en la cual navega el usuario, tomando medidas de latencia a servidores elegidos de manera pseudo-aleatoria, distribuidos en la región de LACNIC (servidores *Speedtest* de *Ookla)*
4. El medidor luego de ejecutar las medidas, genera un archivo en formato XML que envía al servidor de aplicaciones mediante el comando POST de HTTP, para ser almacenado en la base de datos de Simón. El App server interpreta el XML para enviarlo a la base de datos.
5. Se almacena finalmente en la base de datos de Simón

En cuanto al caso 2, se puede ejemplificar el flujo de trabajo de la siguiente manera:

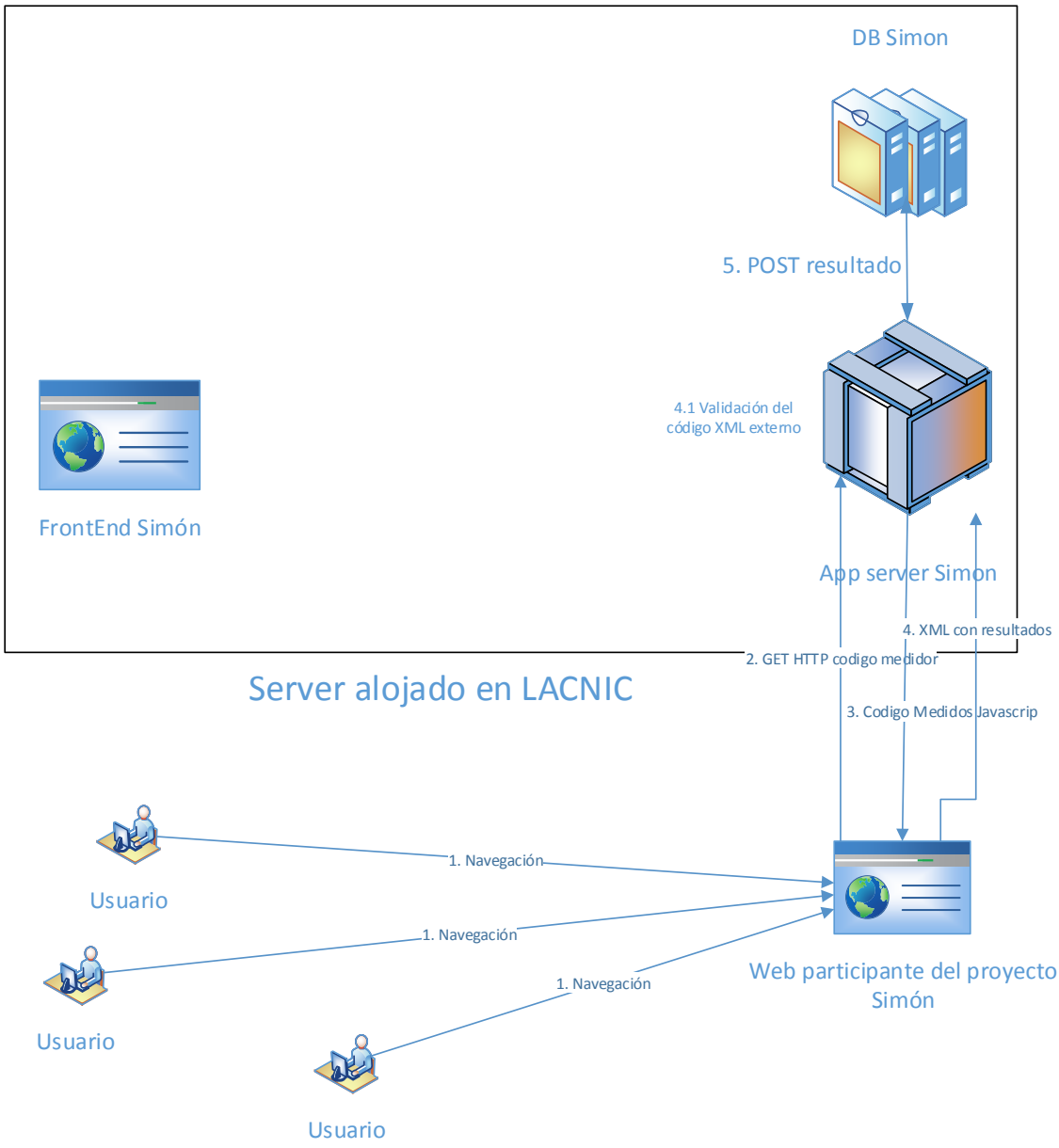


Figura 1.3 – Flujo de obtención de medidas en web externa

Los pasos en el flujo de trabajo son los mismos al caso 1, con la diferencia que las comunicaciones dejan de ser internas al servidor alojado en LACNIC, lo que implica modificaciones en el código contenido en la página de participantes externos y la realización de validaciones del código XML que se envía al servidor de aplicaciones. Esto se hace a través de una función contenida dentro del *framework* llamada *views*. La misma maneja básicamente pedidos y respuestas web, del tipo GET o POST.

Considerando ahora el funcionamiento general del medidor, el mismo se basa en medir el tiempo de respuesta de una consulta GET, de una URL generada específicamente para obtener una respuesta del código 404 del protocolo HTTP, a un servidor particular, elegido previamente de una tabla de la base de datos de Simón.
La medida de tiempo de respuesta será considerada como la latencia de este medidor. Esta medida puede ser configurada para ser realizada más de una vez y a la cantidad de servidores de prueba que se desee. Estas configuraciones pueden ser realizadas solamente por el encargado del proyecto en LACNIC, dependiendo las características de medidas que se desean realizar (ver Capitulo 6 para más información)
El comportamiento general será descrito nuevamente en detalle en las secciones 2.1.4 y 2.2.3.

En cuanto a los datos que se desean obtener, podemos destacar los siguientes campos a ser almacenados en la base de datos (entre otros):

- RTT máximo: máximo valor de latencia a un destino
- RTT mínimo: mínimo valor de latencia a un destino
- RTT mediana: valor de medida, en la posición media del conjunto ordenado de medidas obtenidas en una prueba. A modo de ejemplo:
    - N = cantidad de medidas de un conjunto
    - M = N/2 (redondeo hacia abajo)
    - Si N es impar => RTT mediana = lista[M+1]
    - Si N es par => RTT mediana = (valor[M] + valor[M+1])/2
- RTT promedio: valor promedio de las medidas de un conjunto.
- Desviación estándar: desviación estándar de un conjunto de medidas
    - $\sigma = \sqrt{\frac{\sum_1^n (x_i - \bar{x})^2}{n-1}}$
    - $n = cantidad\ de\ medidas\ del\ conjunto$
    - $\bar{x} = promedio\ de\ las\ medidas\ del\ conjunto$

Estos datos son obtenidos a partir de funciones desarrolladas en el medidor *JavaScript* (ver referencia [36]):

- getMedian(): función para obtener RTT mediana
- getStdDev(): función de obtención de desviación estándar
- getMean(): función para obtener promedio.

A partir de estos valores se pueden generar luego distintas vistas de los datos.
En primer lugar los datos almacenados en la base de datos de Simón pueden ser consultados de forma abierta a través de una API en base a la siguiente URL:

simon.lacnic.net/api/latency/<Country Code>/<Versión de IP>/<Año>/<Mes>

Donde los parámetros entre "<>" son opcionales. A modo de ejemplo se utiliza la siguiente URL:
http://simon.lacnic.net/simon/api/latency/UY/2015

Obteniéndose resultados presentados en formato JSON


```
[
    {
        "as_destination": 28169,
        "as_origin": 6057,
        "ave_rtt": 371,
        "country_destination": "BR",
        "country_origin": "UY",
        "date_test": "2015-10-05 08:12:15+00:00",
        "dev_rtt": 14,
        "ip_version": "4",
        "max_rtt": 404,
        "median_rtt": 368,
        "min_rtt": 343,
        "tester": "JavaScript"
    },
    {
        "as_destination": 8151,
        "as_origin": 6057,
        "ave_rtt": 330,
        "country_destination": "MX",
        "country_origin": "UY",
        "date_test": "2015-10-05 08:12:25+00:00",
        "dev_rtt": 5,
        "ip_version": "4",
        "max_rtt": 343,
        "median_rtt": 330,
        "min_rtt": 320,
        "tester": "JavaScript"
    },
    {
        "as_destination": 53145,
        "as_origin": 6057,
        "ave_rtt": 188,
        "country_destination": "BR",
        "country_origin": "UY",
        "date_test": "2015-10-05 08:12:34+00:00",
        "dev_rtt": 9,
        "ip_version": "4",
        "max_rtt": 198,
        "median_rtt": 193,
        "min_rtt": 158,
        "tester": "JavaScript"
    }, CONTINUA…
```

A su vez los datos pueden ser obtenidos de manera gráfica en la sección “Reportes” dentro de la web de Simón. A continuación se detallan algunos ejemplos en base a medidas desde el mes de enero hasta octubre del 2015, con origen en Uruguay y destino en Brasil:

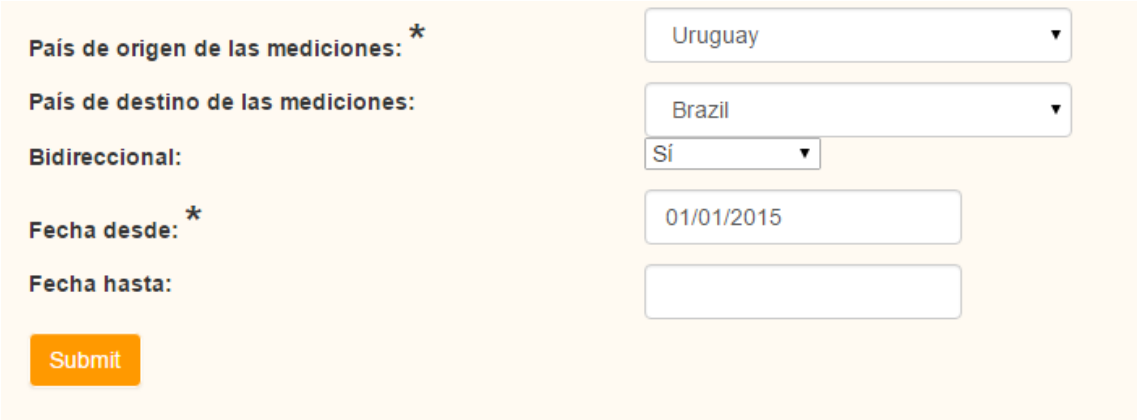


Figura 1.4 – Elección de destinos y fechas de reportes en web Simón

Figura 1.5 – Extracto Tabla de latencias por país en base a Uruguay

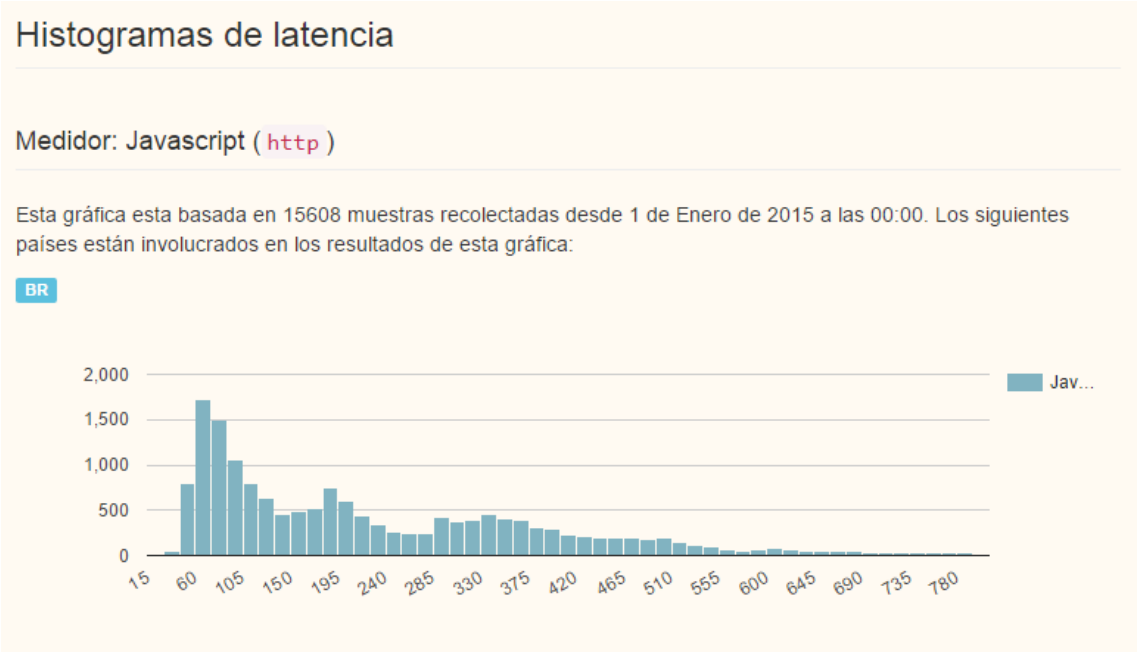


Figura 1.6 – Histograma de medidas entre Uruguay y Brasil medidas Javascript

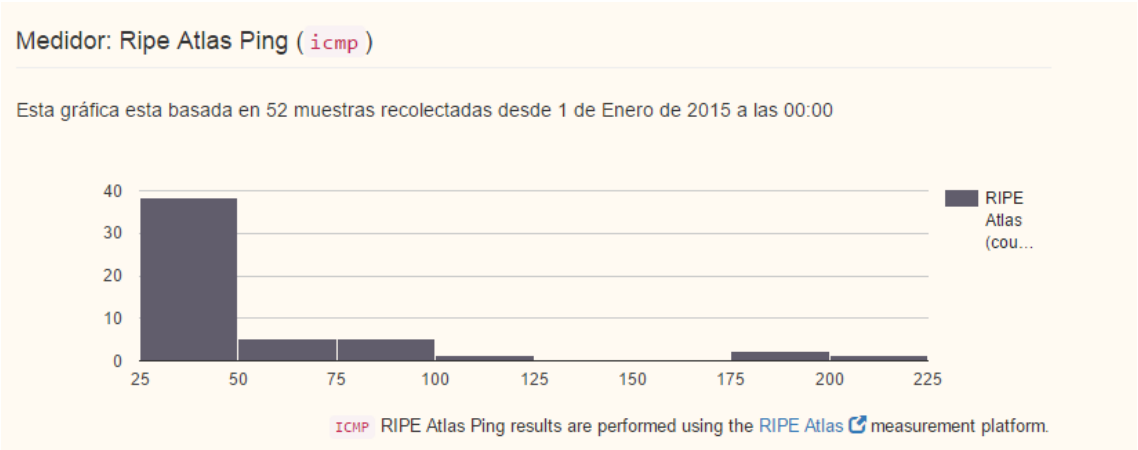


Figura 1.7 – Histograma de medidas entre Uruguay y Brasil medidas RIPE Atlas

Las visualizaciones presentadas en la web de Simón sirven como ejemplo para mostrar cómo se puede manipular la información en base a la obtención de datos a través de la API del proyecto. A su vez buscan ser un incentivo para que la comunidad de LACNIC utilice los datos para generar sus propios contenidos.

1.6 Entorno de trabajo de Simón

Dado que el aporte al proyecto Simón es realizado sobre una base ya consolidada, se consideraron varios caminos, a la hora de definir como se procedería con los scripts y las pruebas en un entorno más reducido y controlado, evitando generar problemas en los ámbitos de producción del proyecto.

Inicialmente todo el trabajo está pensado para ser realizado sobre el sistema operativo y recursos de una laptop. Dadas las características de trabajo de LACNIC, el ambiente ideal es sobre equipos con sistema operativos *MAC OS*, pero teniendo ciertas restricciones iniciales de presupuesto, se decide utilizar un equipo con sistema operativo *Windows*.

A su vez se definen los entornos de programación, IDE (*Integrated Development Enviroment*) por sus siglas en inglés, para el cual se eligió *PyCharm* [12] el cual permite utilizar su interfaz para programar *Python* y *Javascript* indistintamente, sin necesidad de adaptar parámetros especiales. Se utiliza como una opción más amigable a la programación sobre editores de texto plano. *Pycharm* será utilizado en su versión para *Windows*.

En una primera instancia, luego de ser revisados los tutoriales de *Django,* se comienza parte del proyecto, utilizando librerías de *Python* adaptadas para ambientes *Windows.* Luego de realizar algunos scripts, en particular los de la sección tres de este proyecto, se decide descartar la opción de trabajar de esta manera, ya que la interacción con la base de datos y el entorno web de Simón se hace sumamente trabajosa.
Esta decisión fue tomada en gran medida, por las librerías y *plugins* necesarios para el correcto funcionamiento de todas las funcionalidades de Simón.
En este caso particular se destacan las siguientes:

- *Python-dev*: Librería básica de *Python* para programación [13].
- *Libpq-dev:* Librería necesaria para compilar lenguaje C para interactuar con bases de datos Postgres [14].
- *Psycopg*-2: Adaptador de consultas multi-hilos [15].
- *Netaddr:* paquete de manipulación de direcciones IP [16].
- *ixml*: adaptador de XML para *Python* [17]

Por lo tanto se decide recurrir al camino de la virtualización, utilizando máquinas virtuales (VM por sus siglas en inglés) sobre el sistema operativo *Windows*. La opción elegida es *Vagrant*.

La idea de *Vagrant* es proveer VMs ligeras, de rápida configuración a partir de un solo archivo, llamado *Vagrantfile,* que permite ingresar a los ambientes

virtualizados mediante SSH (*Secure Shell*) a la ip de *localhost* (127.0.0.1) en un puerto determinado.
Se brinda un sistema operativo basado en *Unix*, en este caso la versión 12.04 de *Ubuntu,* sin el entorno gráfico, que no es necesario en el proyecto [18] [19].

A su vez *Vagrant* soporta algunas funcionalidades muy útiles para el proyecto. Las más utilizadas son:

- posibilidad de redirigir puertos desde la máquina virtual, al sistema operativo base.
- posibilidad de configurar carpetas compartidas entre la VM y el sistema operativos base.

La VM instalada será soportada en nuestro caso por *VirtualBox* [20].

Para realizar la puesta en marcha de la VM, se necesitan realizar los siguientes pasos:

- Descargar imagen de la VM de la página oficial
- Configurar el acceso por SSH a un puerto especifico, en nuestro caso utilizamos el puerto por defecto 2222 de la configuración de *Vagrant*
- Para comenzar a utilizar la VM, solamente se debe ejecutar el comando "vagrant up" en una consola de comandos de *Windows.* Lo que tendrá como resultado la carga de la maquina con la siguiente detallada en la Figura 1:

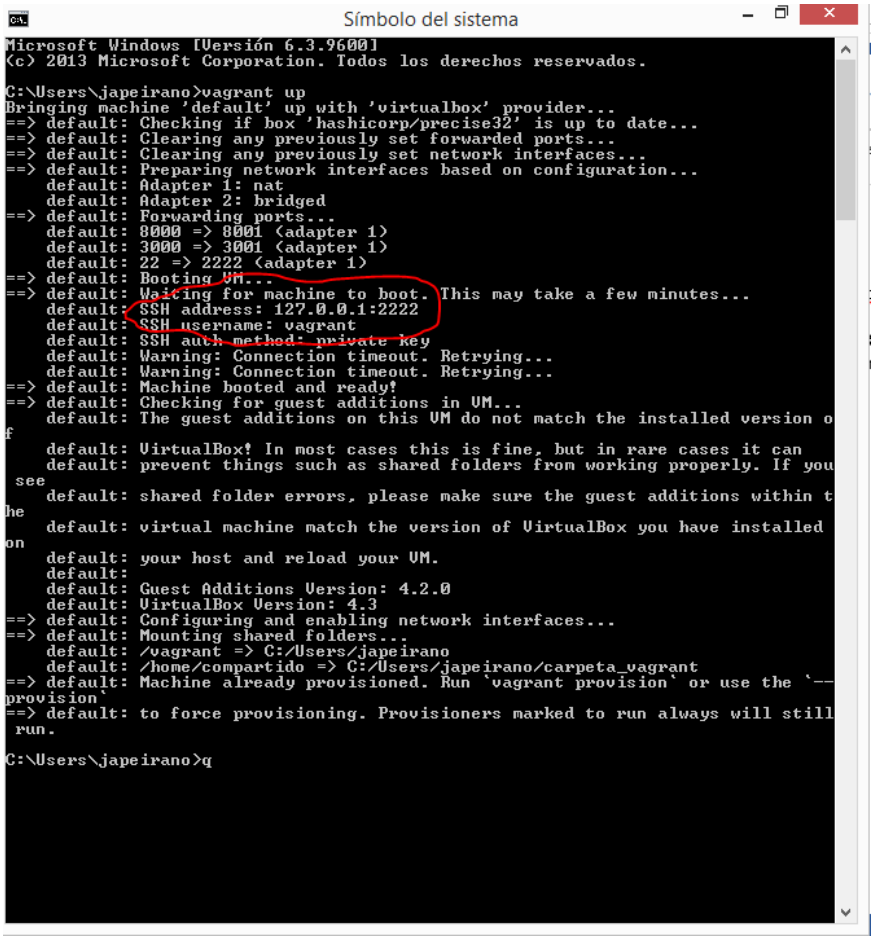


Figura 1.1 – Línea de comando con inicio de *Vagrant*

Luego de tener la VM en condiciones se procede a cargar el código fuente de Simón y todas las librerías necesarias para obtener un ambiente de pruebas lo suficientemente real, que permita realizar las actividades del proyecto.
El código fuente de Simón es cargado en la VM de una manera particular, en una carpeta compartida [21] entre el sistema operativo base *Windows* y el sistema operativo virtualizado *Ubuntu*.
Esta funcionalidad es una capacidad desarrollada en las virtualizaciones *Vagrant*, lo que facilita la edición y programación de los scripts utilizando el IDE *Pycharm.*

Continuando con la instalación, LACNIC provee una copia de la base de datos de Simón utilizada en producción, para para programar y corroborar los resultados de los scripts propuestos.

Finalmente es necesario realizar una redirección de un puerto [22] a elección, para poder observar los cambios realizados en un navegador web sobre el sistema operativo base.
Esta redirección se hace, al igual que para la conexión SSH, a través del archivo *Vagrantfile.* En el caso del proyecto se redirige el puerto 8000 al 8001. De esta manera al ejecutar Simón en la VM en el puerto 8000, podremos ingresar a la página web, ingresando a http://localhost:8001.

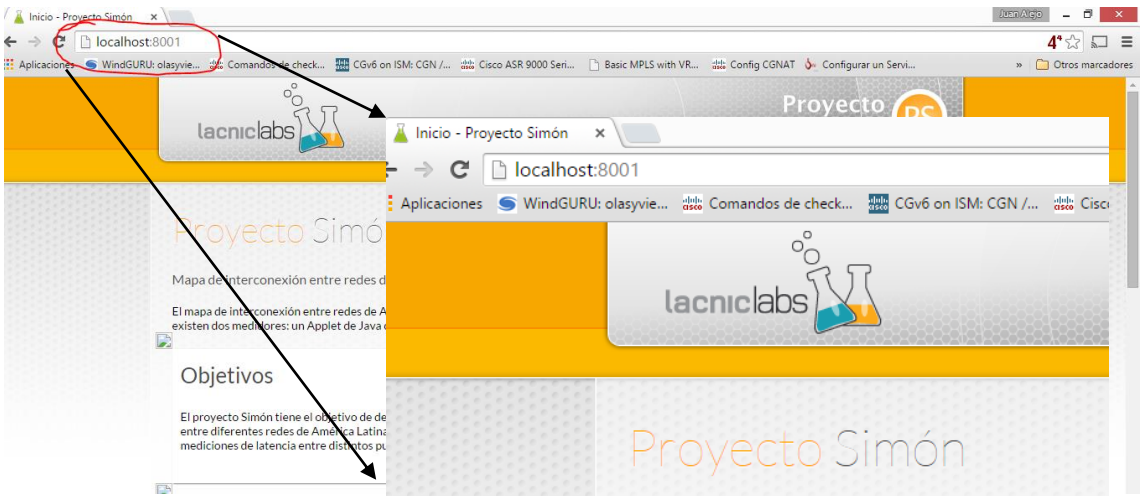


Figura 1.2 – Página web de Simón en entorno de maqueta

Teniendo las herramientas listas, se pueden comenzar con las actividades estipuladas.

Capítulo 2 - Normalización de datos

2.1 Normalización de los resultados

2.1.1 Introducción

Los resultados provienen de diferentes medidores y se recopilan en una base de datos central. El objetivo principal de esta actividad es el estudio y posible implementación de una capa de software que abstraiga las diferentes fuentes de datos y provea un único servicio, donde se pueda consultar por los resultados.

Teniendo en cuenta lo descrito anteriormente, durante el desarrollo de esta sección definiremos *Normalización de datos* como la implementación de software, que permite la manipulación de los resultados en la base de datos, de los diferentes medidores, de manera de presentar un resultado de medición, para una pareja origen – destino (es decir, una medición realizada entre un dispositivo identificado con una dirección IP origen y otro dispositivo identificado con una dirección IP destino) elegido, como un resultado unificado, sin importar con que herramientas se obtuvo la medida.

Actualmente la latencia es mide de varias maneras en Simón:

- ***ping TCP***
  - Utiliza una conexión TCP, levantando un socket y midiendo el tiempo entre los mensajes de sincronización entre el origen y el destino.
- ***NTP***
  - Realizado mediante un aplicativo incluido en el proyecto Simón.
  - Utilizando los servidores NTP distribuidos en la región de influencia de LACNIC
- ***JavaScript***
  - Realizado mediante un aplicativo incluido en el proyecto Simón.
  - Se realiza una consulta *GET* HTTP por un recurso inexistente, esperando un mensaje de error HTTP, en general el 404 y se mide el tiempo de respuesta.
  - Los servidores de destino utilizados son los de *SpeedTest*, elegidos de manera aleatoria.
- **Icmp** (RIPE, CAIDA)
  - Tanto RIPE NCC como CAIDA utilizan el protocolo ICMP para realizar las pruebas de latencia.

Para realizar el estudio se comparan algunas características de las medidas:

- Proceso de obtención de medidas

- Comportamiento de los paquetes en una red.

A partir de la comparación se busca encontrar relaciones entre las diferentes herramientas de obtención de datos y concluir si es posible o no generar una capa de normalización para todas las medidas.

2.1.2 Medidor TCP

TCP es un protocolo de capa de transporte, en relación al modelo OSI, que posee características particulares en el establecimiento de una conexión.
La misma se establece a partir de un procedimiento de tres vías [23], lo que implica tres segmentos TCP enviados antes de efectivamente enviar datos. En la Figura 2.1.1 se puede ver el proceso total de establecimiento de una conexión:

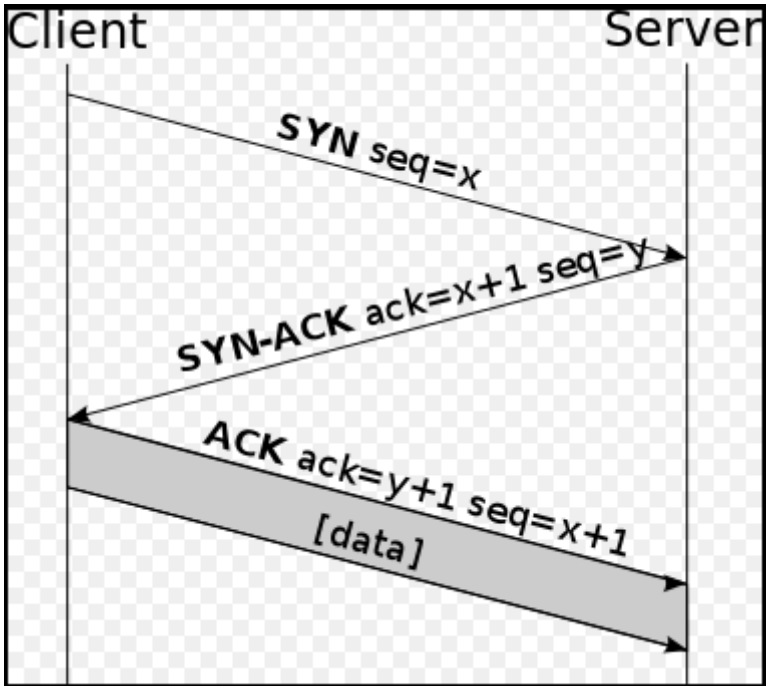


Figura 2.1 – Establecimiento de conexión TCP de tres vías

El mensaje inicial establecido por el cliente, **SYN**, es enviado mediante un segmento especificando el numero inicial de secuencia, que se utiliza para ordenar los paquetes enviados luego de establecida la conexión.
Luego el servidor puede aceptar o rechazar la conexión mediante mensajes de **SYN-ACK** o **RST** respectivamente.

Para el caso del medidor **ping TCP**, se realizan pruebas a los puertos 80 (web) y 53 (DNS), dependiendo del tipo de punto de test que se obtenga de la base de datos, midiendo el tiempo de respuesta que demora el mensaje desde el servidor al cliente.

El código que se utiliza para realizar la prueba está basado en *Java* y se compone básicamente de 4 pasos:

1. *long ti=System.currentTimeMillis();*
   Se obtiene la hora al momento de realizar la prueba en milisegundos
2. *socket = new Socket();*
   Se crea el socket (inicializa)
3. *socket.connect(address);*
   Se introduce la IP de destino
4. *long dt=System.currentTimeMillis() - ti;*
   Se mide nuevamente la hora en milisegundos y se le resta la hora al inicio para medir el tiempo de respuesta.

Para este tipo de pruebas se utilizan servidores *SpeedTest,* aproximadamente 400 distribuidos en la región.

Para hacer uso de este medidor, cada cliente debe obtener el código *java* y se debe ejecutar de forma manual ya que se solicitan permisos de ejecución al momento de realizar la prueba.

2.1.3 Medidor NTP

El protocolo NTP [24] se utiliza para realizar sincronización de relojes en redes IP y utiliza UDP como protocolo de transporte (capa 4 de modelo OSI). Este protocolo funciona dentro de una arquitectura llamada de "estratos de reloj", donde el estrato 0 son los relojes atómico o relojes basados en GPS, el estrato 1 toma la información de sincronización del estrato 0, lo mismo hacen los equipos del estrato 2 con el 1 y así sucesivamente.

Para consultar a un servidor que brinda servicios de sincronización NTP, es necesario enviar un paquete con un formato particular y determinados campos configurados.
El medidor NTP implementado en la plataforma de Simón, está desarrollado en lenguaje *Java,* posee una clase que permite construir estos paquetes y definir los parámetros del mismo para ser enviado a los servidores NTP en la red. Algunos de los parámetros más importantes definidos son:

- Versión de NTP utilizada: 3 (versión compatible solo con IPv4)
- Estrato de reloj a ser alcanzado: indefinido
- Referencia de reloj en el estrato 0: indefinida (no se pretende llegar a ningún servidor en particular)
- Pruebas por cada servidor: 5 (una cada 16 segundos)

Más atributos pueden ser definidos, pero pueden no considerarse relevantes en el manejo del medidor.

El código que finalmente decide la medida de latencia que se va a tomar es el siguiente, en el cual se mide el tiempo que demora un paquete NTP entre su envío y recepción

Transmisión de paquete NTP:

1. *DatagramPacket packet = new DatagramPacket(buf, buf.length, address, 123);* [Se crea una instancia de un paquete del tipo NTP para enviar al servidor de prueba]

   [Código intermedio de adaptación del paquete NTP]

2. *long ti=System.currentTimeMillis()*; [Se inicia el tiempo para medir el RTT]
3. *socket.send(packet);* [Se envía el paquete al servidor NTP]

Recepción de paquete NTP:

4. *packet = new DatagramPacket(buf, buf.length);* [Se crea una instancia de un nuevo paquete del tipo NTP para recibir la respuesta del servidor de prueba a través de un socket.]
5. *socket.receive(packet);* [Se obtiene el paquete NTP]

6. *rtt=System.currentTimeMillis()-ti*; [Se obtiene el tiempo entre el envío y la recepción de una solicitud NTP]

De manera similar al caso del medidor TCP, se utilizan aproximadamente 200 servidores de sincronismo NTP.
Al igual que en el medidor anterior, cada cliente debe obtener el código y se debe ejecutar de forma manual.

2.1.4 Medidor JavaScript

A diferencia de los medidores anteriores, el medidor de *JavaScript* se encuentra implementado en una capa superior, en la de aplicación, ya que utiliza los mensajes de HTTP como controlador de tiempos de respuesta de los servidores consultados.

El medidor está basado en la realización de consultas a través de una librería de *JavaScript* llamada *JQuery* [25], que permite, entre otras cosas, interactuar con páginas web.
Esta herramienta permite realizar una consulta a una url en particular, utilizando la función GET de HTTP. La consulta GET es realizada a los mismos servidores que en el caso del medidor TCP.

Primeramente se realiza un chequeo de que la IP del servidor, corroborando que se encuentre en línea. Luego se genera una url basada en la IP, apuntando a un recurso inexistente, generado por el número aleatorio. Un ejemplo de la url es el siguiente:

http://[ip_servidor]/[nro_aleatorio]

A partir de esa información se mide el tiempo entre que la consulta es realizada y que el mensaje estándar de HTTP llega nuevamente a la maquina desde la cual se está realizando las consultas.

Este medidor es el que ha sido utilizado en mayor medida por LACNIC en los últimos años, ya que puede ser ejecutado de manera transparente para el usuario final y no es necesario habilitar permisos de ejecución de ningún tipo. A su vez se puede acceder al código para integrarlo a la página web como colaboradora del proyecto y desde ahí generar medidas. Esta última opción permite generar medidas del tipo *mesh* desde distintos puntos al mismo tiempo [26] como fue mencionado en la sección 1.5.

Se puede observar la porción de código utilizada para realizar la medida de latencia a continuación, basado en una *JQuery* (consulta asíncrona):

```
$.jsonp({
        type : 'GET',
        url : url,
        dataType : 'jsonp',
        timeout : SIMON.latencyTimeout,
        xhrFields : {
                withCredentials : true
        },

        beforeSend : function(xhr) {

                if (xhr.overrideMimeType)
                        xhr.setRequestHeader("Connection", "close");

        },

        error : function(jqXHR, textStatus) {
                if (textStatus == 'timeout') {
                        testPoint.results.push('timeout');

                } else {
                                /*
                                 * If there is an error and the site is up, we can suppose
                                 * it is due to 404
                                 */
                        rtt = (+new Date - ts);
                        testPoint.results.push(rtt);
                        SIMON.printr("Measuring latency to " + testPoint.ip + " - "+ rtt
                + " ms");
                        SIMON.after_each(rtt);
                }
```

A continuación se detalla la utilización de los campos resaltados:

- Consulta GET HTTP
- "url": dirección web generada previamente para obtener adrede un mensaje de respuesta HTTP 404
- "ts" es la hora del día en milisegundos, al momento de hacer la prueba
- "textStatus" devuelve el código del mensaje HTTP

2.1.5 Medidor ICMP

Durante el transcurso del siguiente capítulo se desarrollarán más las características de los medidores de RIPE y de CAIDA, por lo que en esta etapa el foco está dispuesto en las características de las medidas obtenidas.

Tanto los medidos de CAIDA como los de RIPE utilizan ICMP para realizar las pruebas de latencia. Este protocolo puede ser considerado como parte de la capa de red (3) del modelo OSI, ya que su fin no es el de transporte como TCP o UDP, sino el de control de conectividad de las redes IP.

Las aplicaciones utilizadas para realizar las pruebas son *ping* y *traceroute.* Ambas herramientas tienen un funcionamiento conocido por dentro del ámbito de las telecomunicaciones, por lo que no se entrará en detalle de las mismas.

Algunos aspectos importantes de estas herramientas son:

- La facilidad en su utilización
- Disponibilidad en todos los ambientes y sistemas operativos
- Confiabilidad
- Sus funcionalidades pueden ser filtradas en los *routers,* para evitar ataques de denegación de servicio (filtrado de paquetes ICMP).

2.1.6 Comparaciones

En la presente sección se busca comparar los medidores pertenecientes al proyecto Simón y establecer la posibilidad, de aplicar una capa de abstracción que normalice los resultados de todos ellos.

En primer lugar se puede examinar la manera de funcionar de cada medidor.
Tanto los medidores de TCP como el NTP, utilizan herramientas dentro de la capa de transporte. En cambio el medidor *JavaScript* utiliza la capa de aplicación y los medidores ICMP la capa de red. Por lo tanto se puede considerar que los medidores pertenecen a 3 grupos distintos, según la capa en la cual se encuentra su accionar, teniendo en cuenta el modelo OSI.

A su vez podemos agrupar los medidores por su facilidad de uso. Para el caso de *JavaScript* la utilización de la herramienta es extremadamente fácil, ya que solamente se debe ingresar a la página web del proyecto Simón (o una página colaboradora) para generar una medida.
Para el caso de CAIDA y RIPE, las medidas son recabadas con scripts que serán detallados durante el documento, por lo que no está dentro del alcance de esta sección si las medidas se realizan de manera sencilla o no.
Finalmente para el caso de los medidores TCP y NTP, las herramientas de medición son más complejas de utilizar, ya que implican manejo con nivel de *administrador* por parte de los usuarios para ejecutar correctamente los scripts, lo que puede llegar a ser perjudicial para el objetivo del proyecto, que es recolectar la mayor cantidad de mediciones posibles.

Por otro lado el comportamiento de la información enviada por cada medidor puede diferir durante su trayecto por Internet. Para el caso de TCP y NTP, los segmentos no debería de verse afectados en su tránsito por la red más allá de los efectos que pueda genera determinados algoritmos de balanceo de carga en los *routers*. En cambio los paquetes ICMP pueden ser manejados con una prioridad inferior en una red, por lo que esto podría afectar la medida de latencia, en un factor difícil de cuantificar.
Para el caso de los resultados de latencia obtenidos mediante el medidor de *JavaScript,* pueden verse afectados por la cantidad de consultas que posee el servidor al cual se está haciendo la consulta, ya que en caso de que el servidor se encuentre muy “ocupado”, la respuesta puede no reflejar el estado de la red sino el estado del servidor en ese momento. Este factor es también difícil de cuantificar.

Finalmente habiendo revisado las características de los medidores y de tener en cuenta el beneficio que puede aportar una capa de normalización en las medidas, se decide no realizarla.

Algunos de los factores que inciden en la decisión son los siguientes:

- Los medidores TCP y NTP son complejos de utilizar, por lo que los encargados del proyecto Simón buscan disminuir su utilización, la cual es poca en estas etapas del proyecto.
  A pesar de que al día de hoy los resultados recabados por estos medidores tienen un peso importante en el total de medidas, se espera evolucionar con el medidor de *JavaScript*, tanto en alcance como en facilidad de uso, haciendo que los resultados de los medidores TCP y NTP no tengan relevancia.
- Las medidas que son de interés, son las obtenidas de las bases de datos de CAIDA y RIPE. Estas medidas se basan en aplicaciones diferentes al medidor *JavaScript*, por lo que se decide mantenerlas como recursos separados, para comparar el efecto de una u otra aplicación, permitiendo tener visiones desde diferentes ángulos, de un mismo problema.

En las secciones de resultados y conclusiones se detallarán aún más algunos de los aspectos que motivaron esta decisión.

2.2 Normalización de los navegadores y sistemas operativos

2.2.1 Introducción

Dentro del proyecto, desarrollado previamente, existe un medidor *Javascript* que se ejecuta en los navegadores de los usuarios al ingresar a la página del proyecto (o a cualquier página) http://simon.lacnic.net ya que el mismo se encuentra embebido en el código de la página mencionada.
Los resultados de este medidor dependen de la performance del sistema operativo, el navegador y la forma mediante la cual el navegador ejecuta el medidor, por lo que los resultados generados por un navegador pueden ser diferentes de los resultados generados por otro.
La diversidad de performance de los navegadores es un problema conocido y no existe una forma sencilla de abstraerse de la combinación navegador web - sistema operativo en el que pueda estar corriendo el script, por lo que es necesario realizar mediciones empíricas, que permitan obtener aproximaciones a la influencia de estos elementos en las medidas de latencia.

Durante el desarrollo de esta actividad se define *Normalización de navegadores y sistemas operativos* como como la implementación de software, que permite la manipulación de los resultados en la base de datos, obtenidos mediante diferentes sistemas operativos y navegadores, de manera de presentar un resultado de medición, para una pareja origen – destino elegido, como un resultado unificado, sin importar con que sistema operativo y navegador se realizó.

Los navegadores y sistemas operativos a ser analizados son los siguientes:

- Sistemas operativos: *Windows XP, Widows7, Windows 8.1, MacOSX*
- Navegadores: *Firefox, Chrome, Safari, Opera, Internet Explorer(IE)*

La elección de los mismos fue realizada en base a restricciones establecidas en los ambientes de prueba y en análisis de *marketshare* de los sistemas operativos y navegadores en el mercado latinoamericano, el cual será desarrollado en profundidad en las siguientes secciones.

Finalmente se pretende tomar una dupla sistema operativo y un navegador como medidas “base”, para normalizar el resto de las mediciones a la dupla elegida. Esta dupla también es elegida en base al análisis de *marketshare* mencionado anteriormente.

2.2.2 Definición arquitectura de maqueta

Durante la planificación inicial de esta etapa se consideró el siguiente escenario de pruebas:

- Configurar máquinas virtuales (del tipo *VirtualBox*) a nivel local con los sistemas operativos base necesarios, de manera de realizar las pruebas del medidor *Javascript* en un solo equipo, homogeneizando el ambiente de pruebas
- Instalar los navegadores a utilizar en las pruebas
- Modificar el medidor para realizar un promedio de mil mediciones a un solo destino fijo

Luego este escenario fue descartado por el tiempo que insume la instalación de los diferentes sistemas operativos, licencias de pruebas e instalación de los elementos necesarios en cada máquina virtual para el correcto desempeño de las mismas.

Finalmente se optó por un escenario de pruebas del tipo cliente-servidor utilizando los siguientes servicios:

1. *Browserstack* [27] como cliente
2. Máquina virtual *Vagrant*, con sistema Simón configurado a modo de servidor
3. Script de medidor *Javascript* modificado
4. Script basado en *Selenium* para automatización de pruebas

Browserstack es una herramienta online que permite realizar pruebas con múltiples sistemas operativos y navegadores a demanda. La misma permite levantar un sistema operativo y un navegador a elección para realizar pruebas de páginas web.

Esta herramienta posee variantes de funcionamiento, que permiten realizar distintos tipos de pruebas:

- *Live*: se solicita un sistema operativo base y un navegador en vivo para realizar una prueba de una url en particular
- *Automate*: Mediante una integración se puede realizar una conexión a los servidores de *Browserstack* para realizar una prueba automatizada, donde se definen las acciones a realizar en la web (urls, movimiento de puntero y barra de desplazamiento, etc.)

La integración mencionada puede realizarse mediante la plataforma *Selenium*, una herramienta de código abierto que permite realizar testeo de páginas web de manera automatizada. En esta sección se utiliza para desarrollar un script con las diferentes combinaciones de la dupla sistema operativo – navegador, previamente definidas.

A continuación y como fue detallado anteriormente, luego de definidas las herramientas técnicas, es necesario realizar un análisis de las pruebas a realizar, las duplas sistema operativo-navegador y la dupla que se define como base para realizar la normalización de las medidas.

Para realizar este análisis se tuvieron en cuenta dos premisas fundamentales:

1. Restricciones en las herramientas tecnológicas.
2. *Marketshare* de sistemas operativos y navegadores en la región de Latinoamérica y el Caribe.

Desde el comienzo, *Browserstack* restringe el nivel de pruebas ya que permite realizar pruebas en los siguientes sistemas operativos y navegadores:

- Sistemas operativos: *Windows XP, Widows7, Windows 8.1, MacOSX*
- Navegadores: *Firefox, Chrome, Safari, Opera, Internet Explorer*

Quedando el sistema operativo Linux y todas sus variantes, inicialmente, por fuera de las pruebas a realizar.

Esta situación hace que sea necesario tener en cuenta cual es la utilización de los sistemas operativos Linux en la región de América Latina y por lo tanto, cuál es su peso en las medidas de latencia del sistema, ya que, sí la utilización del mismo es muy grande, las pruebas y la normalización no se ajustarán a la realidad de la región.

Para analizar esta situación se tienen en cuenta los dos servicios de análisis de tráfico web más conocidos:

- StatCounter [28]
- NetMarketshare [29]

Estos servicios realizan operaciones de *tracking* del tráfico que circula en internet, de manera de obtener información de los sistemas operativos y los navegadores que se utilizan para ingresar a una página en particular.

Ambos servicios poseen metodologías de medición diferentes, que influyen básicamente en los porcentajes de utilización de navegadores, pero no así en lo que respecta a sistemas operativos.

Inicialmente observamos los porcentajes de ambos medidores para el caso de los sistemas operativos:

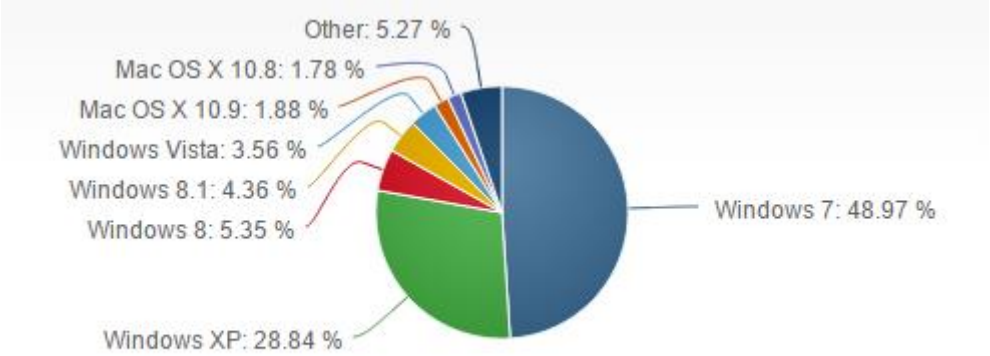


Figura 2.2 – Utilización de sistemas operativos Netmarketshare [30]

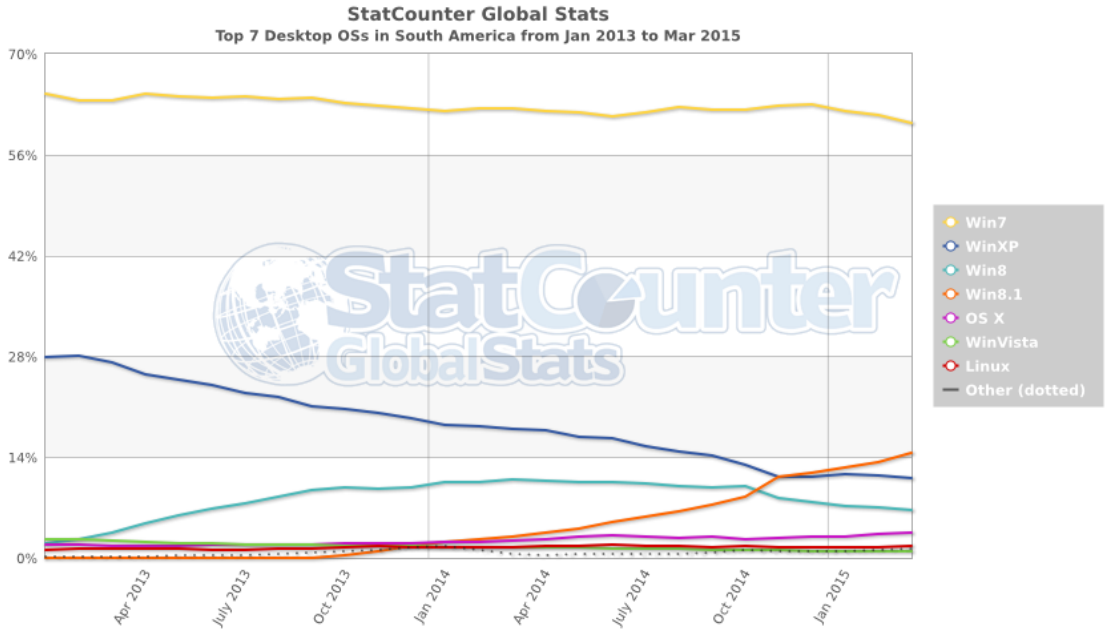


Figura 2.3 – Utilización de sistemas operativos Statcounter [31]

Dado que no se observa un detalle del *marketshare* del sistema operativo Linux, se recurre a los valores mediante los cuales se construyen estas gráficas, que pueden ser obtenidos también de las páginas web de los servicios. De ahí se obtienen las siguientes tablas de datos:

Operating System	**Total Market Share**
Windows 7	48.97%
Windows XP	28.84%
Windows 8	5.35%
Windows 8.1	4.36%

Windows Vista	3.56%
Mac OS X 10.9	1.88%
Mac OS X 10.8	1.78%
Linux	1.46%
Mac OS X 10.6	1.32%
Mac OS X 10.7	1.19%
Mac OS X 10.10	0.65%
Mac OS X 10.5	0.29%
Windows NT	0.15%
Mac OS X 10.4	0.07%

Tabla 2.1 - Utilización de sistemas operativos Netmarketshare

Date	Linux		Date	Linux
2013-01	1.04		2014-03	1.44
2013-02	1.27		2014-04	1.62
2013-03	1.32		2014-05	1.69
2013-04	1.23		2014-06	1.81
2013-05	1.24		2014-07	1.68
2013-06	1.16		2014-08	1.6
2013-07	1.15		2014-09	1.54
2013-08	1.24		2014-10	1.67
2013-09	1.25		2014-11	1.45
2013-10	1.55		2014-12	1.44
2013-11	1.72		2015-01	1.48
2013-12	1.53		2015-02	1.53
2014-01	1.45		2015-03	1.63
2014-02	1.45		Mean	1.4511111

Tabla 2.2 - Utilización de sistemas operativos StatCounter

Se puede apreciar en los valores resaltados, que la porción de mercado que ocupa Linux es lo suficientemente baja, para considerar que el error que se pueda obtener en las medidas y posterior normalización, no se verá significativamente afectado por la falta de generación de duplas sistema operativo- navegador con base en Linux. Esto hace que las *Browserstack* sea una herramienta útil para la realización de las mediciones.

Luego de realizado el análisis desde el punto de vista de los sistemas operativos, es necesario definir cuál sería la dupla sistema operativo – navegador. Nuevamente recurrimos los servicios de análisis de tráfico web para definir cuál sería la dupla más acertada. En este caso la decisión se torna más compleja, ya que los servicios de análisis que se toman como referencia, tienen valores muy disimiles en cuanto a la utilización de navegadores web, los cuales se muestran a continuación.

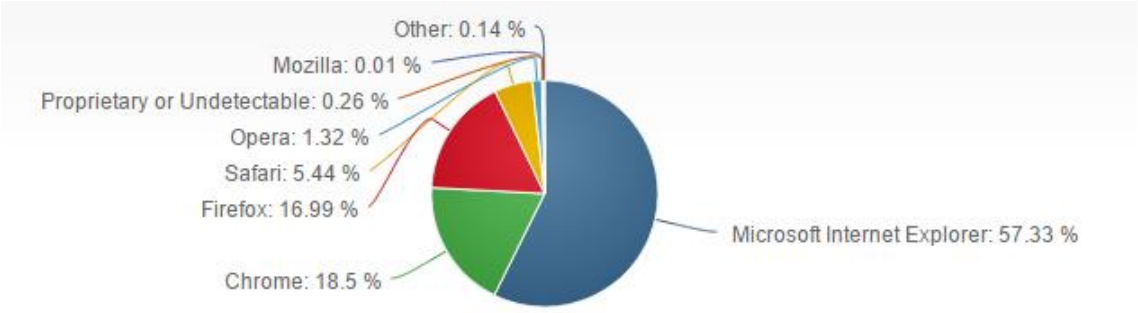


Figura 2.4 – Utilización de navegadores Netmarketshare [32]

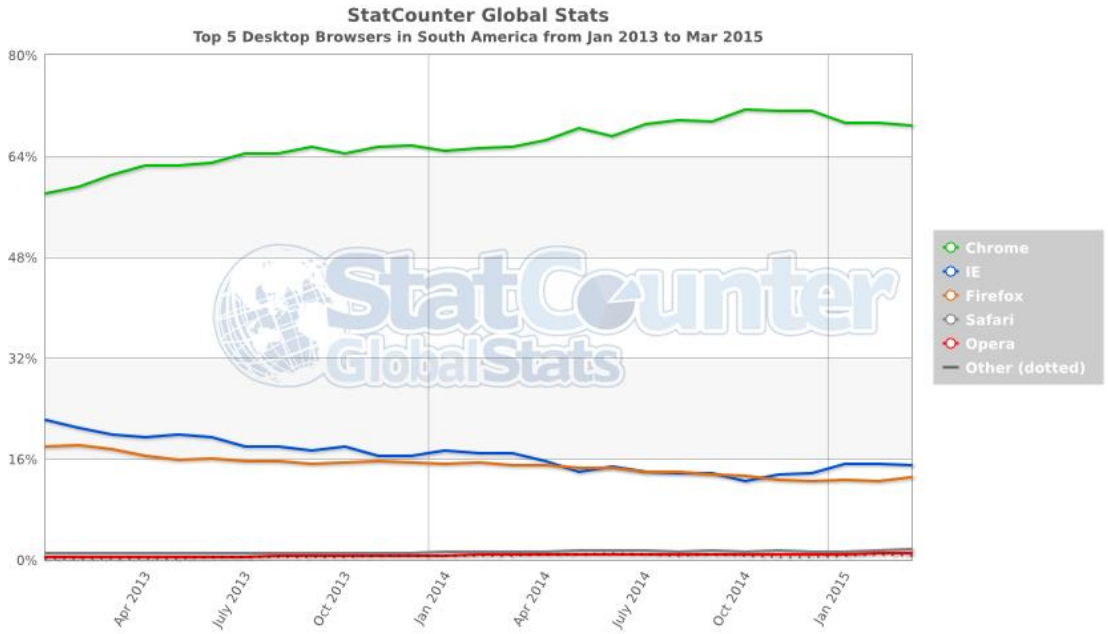


Figura 2.5 – Utilización de navegadores StatCounter [33]

A simple vista se observa claramente una diferencia notoria, en los que respecta las medidas de los navegadores *Internet Explorer y Chrome*. Yendo ahora a los datos numéricos en las tablas 2.3 y 2.4:

Browser	**Total Market Share**
Microsoft Internet Explorer	57.33%
Chrome	18.50%
Firefox	16.99%
Safari	5.44%
Opera	1.32%
Proprietary or Undetectable	0.26%
Mozilla	0.01%
Konqueror	0.01%

Tabla 2.3 - Utilización navegadores Netmarketshare

Date	Chrome	IE		Date	Chrome	IE
2013-01	57.97	22.17		2014-03	65.46	16.8
2013-02	59.08	20.81		2014-04	66.43	15.61
2013-03	60.95	19.8		2014-05	68.41	13.92
2013-04	62.4	19.44		2014-06	67.22	14.88
2013-05	62.57	19.77		2014-07	69.03	13.99
2013-06	62.81	19.38		2014-08	69.65	13.63
2013-07	64.44	18.01		2014-09	69.54	13.79
2013-08	64.47	17.87		2014-10	71.44	12.41
2013-09	65.43	17.28		2014-11	71.07	13.42
2013-10	64.41	17.85		2014-12	71.07	13.75
2013-11	65.5	16.44		2015-01	69.31	15.13
2013-12	65.75	16.36		2015-02	69.14	15.3
2014-01	64.81	17.36		2015-03	68.75	15.02
2014-02	65.15	16.85				

Tabla 2.4 - Utilización de navegadores StatCounter

Se podría decir que las mediciones de cada servicio son prácticamente opuestas, lo que inicialmente hace incierta la elección de un navegador base.

Para encontrar una medida fiable, se recurre a las características que poseen estos servicios al momento de realizar los análisis de tráfico. Esta información la podemos obtener de cada una de sus páginas web, particularmente en las secciones de *FAQ (Frequently Asked Questions)*

A continuación se detallan algunas características mencionadas por cada servicio:

StatCounter [34]:

- 3 millones de páginas de donde recabar información: se incluye un código en la página que permite leer el contenido del *user-agent* (Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.65 Safari/537.36)

- Se realiza poco o nulo manejo de la información recabada: según expresan para evitar metodologías que afecten la interpretación de los datos.

- Basa sus estadísticas en páginas "vistas" (*page views*): cada página que contenga el código de StatCounter alcanzada por un usuario cuenta, no importa un mismo usuario entra a 2 o más paginas dentro del set de páginas a ser medidas.

- Tiene en cuenta las páginas precargadas por algunos navegadores (*prerendering*): Algunos navegadores realizan precargas de páginas (en

particular *Chrome*) para agilizar la apertura de las mismas en caso que el usuario decida entrar, por lo que se generarían entradas "falsas". Según el FAQ de StatCounter, se tiene en cuenta el *prerendering* y se descartan las entradas a páginas precargadas.

NetMarketshare [35]:

- 40 mil páginas de donde recaba información: al igual que StatCounter coloca un código para realizar las mediciones

- Se realiza una manipulación de los datos recabados: se realizan comparaciones con una base de datos de la CIA (*Central Intelligence Agency*) de manera de sopesar variaciones en los porcentajes medidos por país. Aseguran nivelar el porcentaje de utilización por región.

- Basa sus estadísticas en usuarios únicos (*unique-visitors)*: se tiene en cuenta solamente la visita de un usuario a una de las páginas por día.

- Se tiene en cuenta las páginas precargadas por algunos navegadores, al igual que StatCounter.

Las características antes mencionadas, son algunas de las que pueden ser encontradas en las páginas de *FAQ* correspondientes. Inclusive se pueden encontrar comparaciones de cada una de los servicios que se tuvieron en cuenta en este análisis.

Teniendo en cuenta todos estos factores, es necesario tomar una decisión de que navegador utilizar para realizar las normalizaciones, al menos inicialmente.
Se podría decir que ambos sistemas de medición tienen cometidos distintos y debemos elegir el que se ajusta más a las necesidades del proyecto.
Por un lado StatCounter busca saber cuáles son los navegadores más utilizados en el tráfico total de internet (hay más hits de Chrome en las páginas web), esto es, por ejemplo para el caso de *Chrome*, es el navegador con el cual los usuarios más navegan. Un usuario que utiliza *Chrome* se dirige a muchas más páginas web que un usuario que utiliza por ejemplo Internet Explorer, pero son en definitiva 2 usuarios navegando, uno de manera más "intensa" que el otro. Se podría decir que un usuario de Chrome utiliza mucho más su navegador que un usuario de IE por lo que tiene un peso más alto el usuario de Chrome que el de IE.
Por otro lado NetMarketshare tiene en cuenta el usuario que utiliza un navegador en particular. Teniendo en cuenta el mismo ejemplo anterior, un usuario que entra a una página web en el día por IE, cuenta lo mismo que un usuario que paso el día entero navegando con Chrome, ambos cuentan uno. En este caso se podría decir que se mide efectivamente la cantidad de navegadores y de qué tipo que fueron utilizados durante un día dado.

Para el caso del proyecto Simón se decide utilizar como dupla sistema operativo-navegador:

WINDOWS 7 – CHROME

Esta elección se realiza en base al público objetivo del proyecto. En general el mismo se compone de usuarios que utilizan de manera importante internet, por lo que es más factible encontrar más mediciones provenientes del navegador Chrome, que de cualquier otro navegador del mercado. Por lo tanto con la dupla elegida, a priori se deberían realizar menor cantidad de normalizaciones, manteniendo así la mayoría de los datos incambiados.

2.2.3 Configuración y programación de maqueta

Luego de realizados los análisis previos y definido las característica de las pruebas, se comienza con la puesta a producción de la misma, iniciando por la revisión y comprensión del medidor actual utilizado.

El medidor de *JavaScript* se encuentra bajo la denominación *simon_probe_plugin.js*, [36] el cual contiene una serie de funciones básicas para su funcionamiento y otras funciones definidas, pero aún sin ningún propósito particular definidas para futuras modificaciones al script.
En este caso se hace hincapié en las funciones básicas de funcionamiento, ya que se deben realizar modificaciones a estas funciones y no agregar nuevas funcionalidades.

En el código se pueden identificar las siguientes funciones básicas:

- getCountry()
  Consulta getCountry del archivo api_views.py realiza un consulta WHOIS a la IP del usuario del servicio, para obtener el país en el cual se origina la consulta. Para el caso de las pruebas de esta etapa, no será necesario identificar cual es el país de donde proviene la consulta, por lo que será identificado con el código "LH" haciendo referencia a LocalHost

- getMyIPAddress()
  Esta función devuelve la IP del usuario del servicio. Al igual que en el caso anterior, se define la IP de localhost por defecto 127.0.0.1 de manera de guardar los datos en la base como información fácilmente reconocible y claramente diferente del resto de los resultados.

- getTestsConfigs()
  La presente función recaba información de los TestPoint en base al tipo de IP (v4 o v6) que posee el usuario que realiza el test de conectividad. Luego procede a buscar los puntos de test con la siguiente función.

- getPoints()
  Teniendo en cuenta la versión IP del usuario, a través de esta función se solicitan los puntos de test a donde se realizaran las pruebas de latencia. El script se encuentra configurado para solicitar 5 destinos de manera aleatoria para realizar las pruebas. Estos destinos se encuentran en una tabla en la base de datos de Simón y son, como ya fue mencionado anteriormente, servidores de *SpeedTest* de *Ookla* distribuidos en la región de Latinoamérica y el Caribe.

Para el caso de las pruebas realizadas en esta etapa, se agrega un punto te test a la tabla mencionada, con el destino particular www.google.com
A continuación se detalla la salida de la tabla base de datos de puntos de test, correspondiente al servidor agregado:

```
description |  ip_address   |        url
-------------+---------------+-----------------------
 Test_lacnic | 74.125.21.104 | http://www.google.com/
```

Donde la IP 74.125.21.104 corresponde a un servidor de búsqueda de la empresa Google, localizado en Estados Unidos.
Todas las pruebas se realizaran a esta IP de destino de manera fija, modificando el script, de manera que no se tengan en cuenta los servidores de *SpeedTest* ni las elecciones de los mismos de manera aleatoria.

A su vez se modifican funciones secundarias, ya que la función getPoints(), solicita los puntos necesarios a través de una url específica, en la cual se define el tipo de IP (v4 o v6) y la cantidad de puntos de testing.
Esta url, dispara la solicitud a la función

- web_points()

Contenida en el archivo api_views.py, la cual será importante para próximas actividades dentro del proyecto.
Web_points() consulta la base de datos de TestPoints según las especificaciones enviadas en la url. Realizando una modificación a esta consulta se asegura obtener el punto de test 'Test_lacnic', de manera de dirigir todas las pruebas solamente a ese punto.

- siteOnLine()
  Dado que las pruebas realizadas por el medidor JavaScript corresponden a mediciones de latencia mediante consultas HTTP, se realiza un chequeo del estado del punto de test elegido. Dadas las características de nuestras pruebas se eligió un servidor de Google, asegurando que el mismo este siempre activo, por lo que esta función podría obviarse, pero se mantiene activa como herramienta de chequeo al momento de hacer las pruebas.

- saveTestsPoints()
  Los puntos obtenidos en mediante la función getPoints() son guardados para su utilización posterior en los procesos de testing.

- startPointTest()
  Mediante la siguiente función se itera entre los puntos de prueba. Nuevamente, para las pruebas realizadas en esta etapa, la iteración será solamente sobre un punto de test. A su vez controla la iteración de cada test de latencia realizado, estableciendo una suerte de "agenda", guardando en un buffer la cantidad máxima de pruebas a realizar a un destino y luego realizando las pruebas de latencia en periodos definidos de tiempo. Po ejemplo si el máximo de pruebas por punto de testeo son 5, define las 5 pruebas cada un segundo y luego las ejecuta una a una.

- latencyTest
  Esta función compone una de las secciones más importante del script, ya que lleva a cabo el envío de pruebas.
  El test se basa en realizar un pedido GET del protocolo HTTP, el cual apunte a la IP del test point, junto con un agregado de un número aleatorio, de manera de obtener una respuesta con un mensaje estándar del protocolo HTTP, en este caso un mensaje 404.
  El GET se realiza a la siguiente url: http://ip_testpoint/randomNumber
  Luego mide el tiempo que demora en recibirse la respuesta a pedido, luego ese valor se guarda en una lista.
  Finalmente, luego de realizadas todas las medidas de latencia a un punto particular se obtiene la media y al desviación estándar en base a la lista de medidas realizadas y se guarda en la base de datos, junto con otros valores relevantes de la medida.
  Para el caso de las pruebas de esta etapa, el número total de muestras tomadas es de mil (1000), de manera de disminuir lo más posible las desviaciones, obteniendo una medida con la rigurosidad necesaria para que pueda ser considerada como medida base.

Algunas de las funciones mencionadas, fueron modificadas en menor manera, como por ejemplo siteOnLine(), getTestsConfigs(), startPointTest, básicamente para seguir la línea que debería seguir un test normal, tratando de afectar lo menos posible el rendimiento del medidor JavaScript.

Como ya fue mencionado la base de datos de Simón, en particular la tabla de resultados, tiene campos particulares, pero para el caso de las pruebas de esta sección interesan los siguientes:

Column	Type
number_*probes*	integer
dev_rtt	integer
median_rtt	integer
packet_loss	integer
user_agent	text

Tabla 2.5 – Campos DB Simón utilizados

A su vez otro servicio necesario para realizar las pruebas es *Selenium.* Esta herramienta permite definir los atributos y las acciones que deben tener los navegadores y lo sistemas operativos, que se desean utilizar en las pruebas con *Browserstack*.

Los atributos pueden ser encontrados en la página web de *Browserstack* [37]

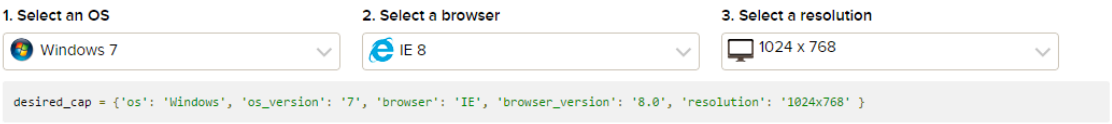


Figura 2.6 – Captura de pantalla de combo box de atributos

La denominación *desire_cap* se refiere a las capacidades deseadas de nuestro sistema operativo y navegador para las pruebas. Dentro de cada combo podemos elegir las siguientes opciones:

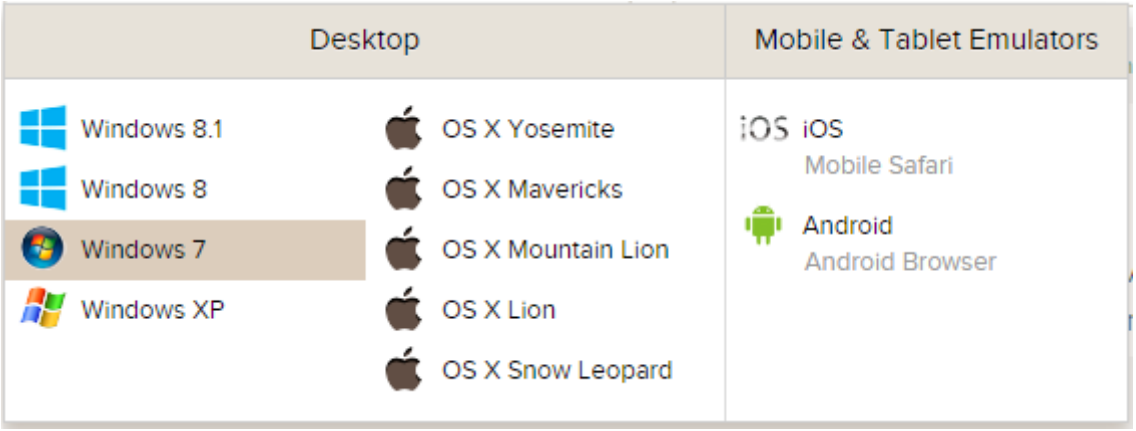


Figura 2.7 – Captura de pantalla combo box sistemas operativos

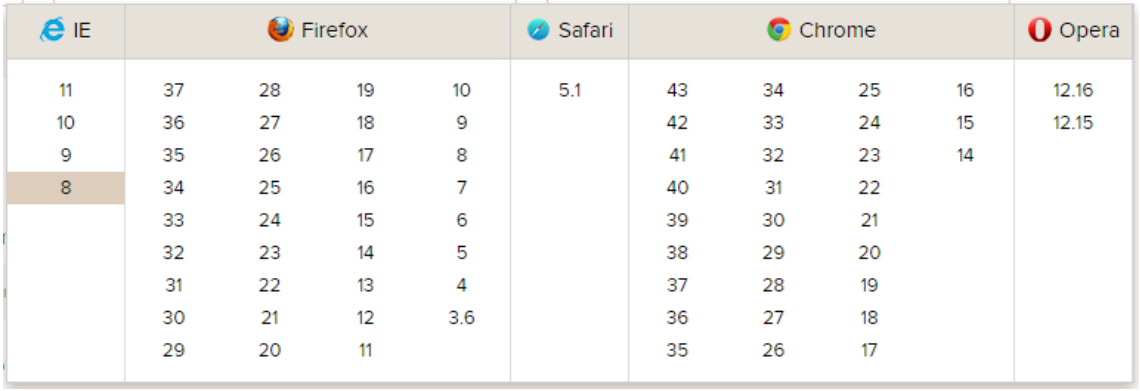


Figura 2.8 – Captura de pantalla combo box navegadores

En todas las pruebas se han elegido versiones de navegadores, anteriores (dos o tres) a la última versión para evitar dificultades por incompatibilidades navegador-sistema operativo.
De cualquier manera en algunos casos particulares se han detectado dificultades, que por problemas de versión, antigüedad de los sistemas o *plugins* particulares, no ha sido posible realizar las pruebas correspondientes

A continuación se detallan las pruebas realizadas:

#PRUEBAS WINDOWS XP

{'browser': 'Chrome', 'browser_version': '32.0', 'os': 'Windows', 'os_version': 'XP','resolution': '1024x768'}

{'browser': 'IE', 'browser_version': '7.0', 'os': 'Windows', 'os_version': 'XP', 'resolution': '1024x768'} #NO FUNCIONA CORRECTAMENTE

{'browser': 'Firefox', 'browser_version': '35.0', 'os': 'Windows', 'os_version': 'XP', 'resolution': '1024x768'}

{'browser': 'Safari', 'browser_version': '5.1', 'os': 'Windows', 'os_version': 'XP', 'resolution': '1024x768'} #NO FUNCIONA CORRECTAMENTE

{'browser': 'Opera', 'browser_version': '12.16', 'os': 'Windows', 'os_version': 'XP', 'resolution': '1024x768'}

#PRUEBAS WINDOWS 7

{'browser': 'IE', 'browser_version': '11.0', 'os': 'Windows', 'os_version': '7', 'resolution': '1024x768'}

{'browser': 'Firefox', 'browser_version': '35.0', 'os': 'Windows', 'os_version': '7', 'resolution': '1024x768'}

{'browser': 'Safari', 'browser_version': '5.1', 'os': 'Windows', 'os_version': '7', 'resolution': '1024x768'}

{'browser': 'Chrome', 'browser_version': '39.0', 'os': 'Windows', 'os_version': '7', 'resolution': '1024x768'}

{'browser': 'Opera', 'browser_version': '12.16', 'os': 'Windows', 'os_version': '7', 'resolution': '1024x768'}

#PRUEBAS WINDOWS 8

{'browser': 'IE', 'browser_version': '11.0', 'os': 'Windows', 'os_version': '8.1', 'resolution': '1024x768'}

{'browser': 'Firefox', 'browser_version': '35.0', 'os': 'Windows', 'os_version': '8.1', 'resolution': '1024x768'}

{'browser': 'Safari', 'browser_version': '5.1', 'os': 'Windows', 'os_version': '8.1', 'resolution': '1024x768'}

{'browser': 'Chrome', 'browser_version': '39.0', 'os': 'Windows', 'os_version': '8.1', 'resolution': '1024x768'}

{'browser': 'Opera', 'browser_version': '12.16', 'os': 'Windows', 'os_version': '8.1', 'resolution': '1024x768'}

#PRUEBAS MAC OS Mavericks

{'browser': 'Firefox', 'browser_version': '35.0', 'os': 'OS X', 'os_version': 'Mavericks', 'resolution': '1024x768'}

{'browser': 'Safari', 'browser_version': '7.0', 'os': 'OS X', 'os_version': 'Mavericks', 'resolution': '1024x768'}

{'browser': 'Chrome', 'browser_version': '39.0', 'os': 'OS X', 'os_version': 'Mavericks', 'resolution': '1024x768'}

{'browser': 'Opera', 'browser_version': '12.15', 'os': 'OS X', 'os_version': 'Mavericks', 'resolution': '1024x768'}

Las pruebas fueron realizadas de manera manual, esto quiere decir, de una a la vez, ya que las características de las mismas hacen que sean en algunos casos susceptibles a fallas. En particular destacamos las siguientes características:

- Para cada prueba, como fue mencionado anteriormente, se realizan mil mediciones, esto hace que cada una tenga una duración aproximada de entre 19 y 21 minutos, dependiendo de la velocidad de inicio de la dupla sistema operativo – navegador elegida.
- Es necesario realizar una conexión segura entre *Browserstack* y el servidor local, en nuestro caso la máquina virtual *Vagrant*
- Ciertas pruebas, como por ejemplo sistemas operativos de una empresa y navegadores de otra, pueden generar desconexiones, errores inesperados o incompatibilidades de software.
- *Browserstack* funciona de manera que, si no hay actividad del navegador durante 90 segundos, se considera que la sesión ha expirado, por lo que es necesario programar el código de *Selenuim* para que realice ciertas actividades durante el tiempo que dura la prueba.

Estos puntos son importantes ya que condicionan la automatización de las pruebas, dado que cada una debe ser controlada de manera de evitar errores o resultados equivocados.

A continuación se detalla un pseudocódigo de la herramienta utilizada para cada prueba [38]:

Conexión a browserstack
Envío capacidades deseadas
GET (url home proyecto Simón para pruebas de Javascript)
For i en 15
    Muevo barra desplazamiento de navegador
    Espero 80 segundos
Termino prueba

Se utiliza una iteración de 15 instancias para cubrir la totalidad de la prueba:

- 1000 pruebas de latencia (a una prueba por segundo) = 17 minutos aprox.
- Inicio de sistema operativo y navegador = 1 minuto aprox.

- Tiempo pruebas = 18 minutos = 1080 segundo
- Tiempo iteraciones = 15 iteraciones de 80 segundos = 1200 segundos

Se mantiene una guarda de aproximadamente 1 minuto, en caso de que las máquinas virtuales remotas de *Browserstack* no respondan de manera adecuada.

A continuación se muestra algunas capturas de pantalla de las pruebas realizadas:

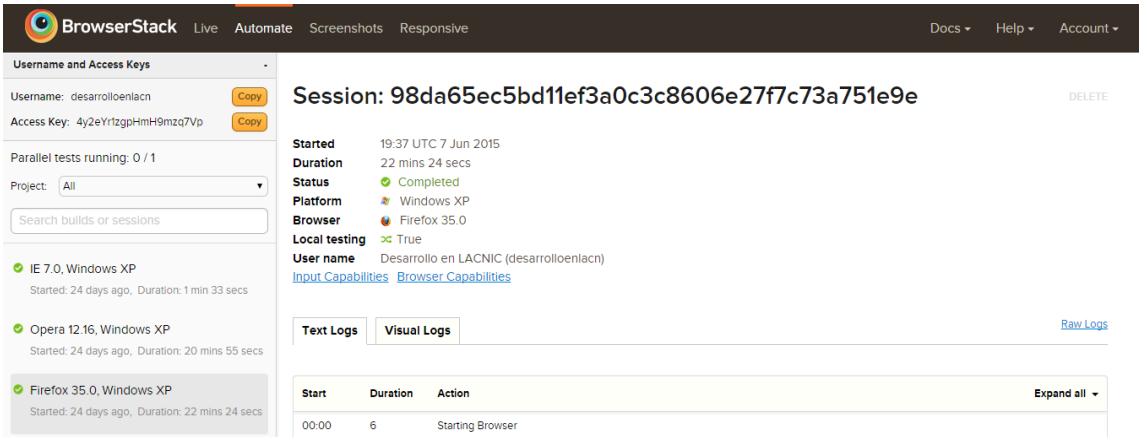


Figura 2.9 – Captura de pantalla prueba exitosa

En la Figura 2.8 se puede apreciar el sistema operativo y el navegador utilizados. A su vez se observa el tiempo de duración de la prueba (22min 24 seg) y el estado de la misma. En este caso fue exitosa.

En la Figura 2.9 se puede apreciar una prueba no exitosa, con un *log* visual, que puede ser utilizado para detectar los errores en la prueba.

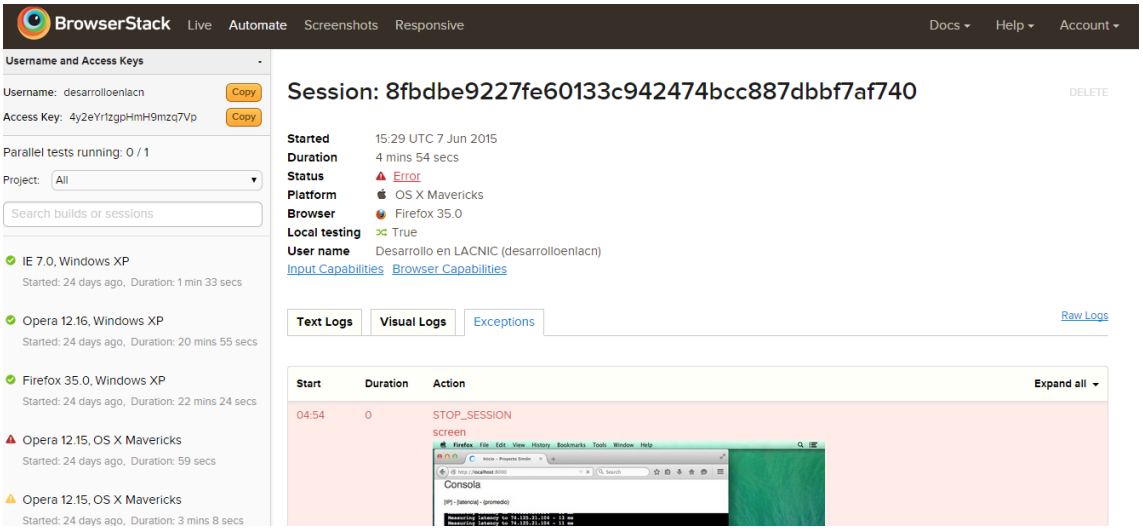


Figura 2.10 – Captura de pantalla prueba no exitosa

Seguidamente se detallan las pruebas realizadas para cada uno de los ambientes:

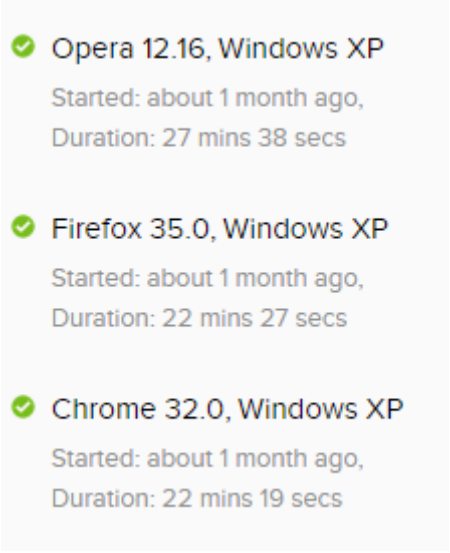


Figura 2.11 – Pruebas Browserstack

Se comienzan las pruebas con Windows XP, teniendo problemas con el navegador Safari, en todas sus versiones y con IE, en el cual el script no se ejecutaba correctamente

Luego con Window 7 todos lo navegadores se comportan de la manera esperada

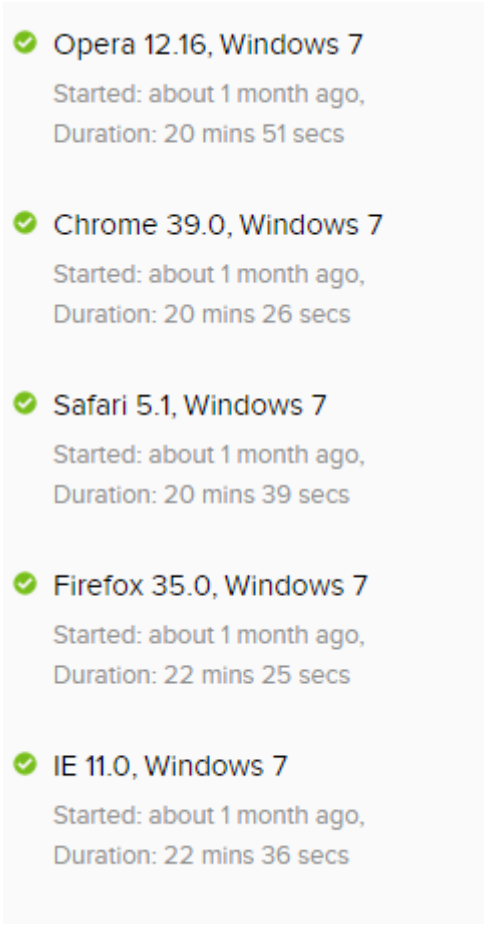


Figura 2.12 – Pruebas Browserstack

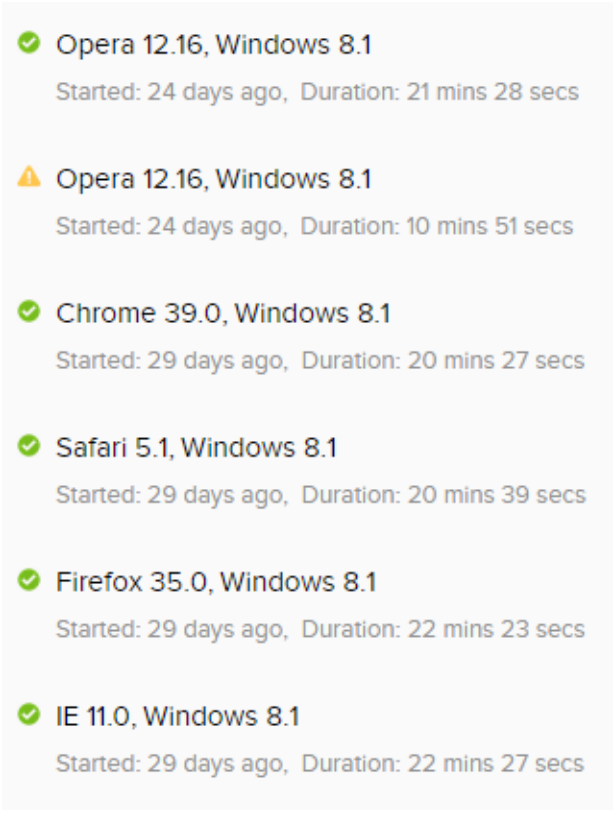


Figura 2.13 – Pruebas Browserstack

Siguiendo con Windows 8.1, existen algunos inconvenientes con el navegador Opera, pero luego de repetida la prueba se obtiene el comportamiento esperado.

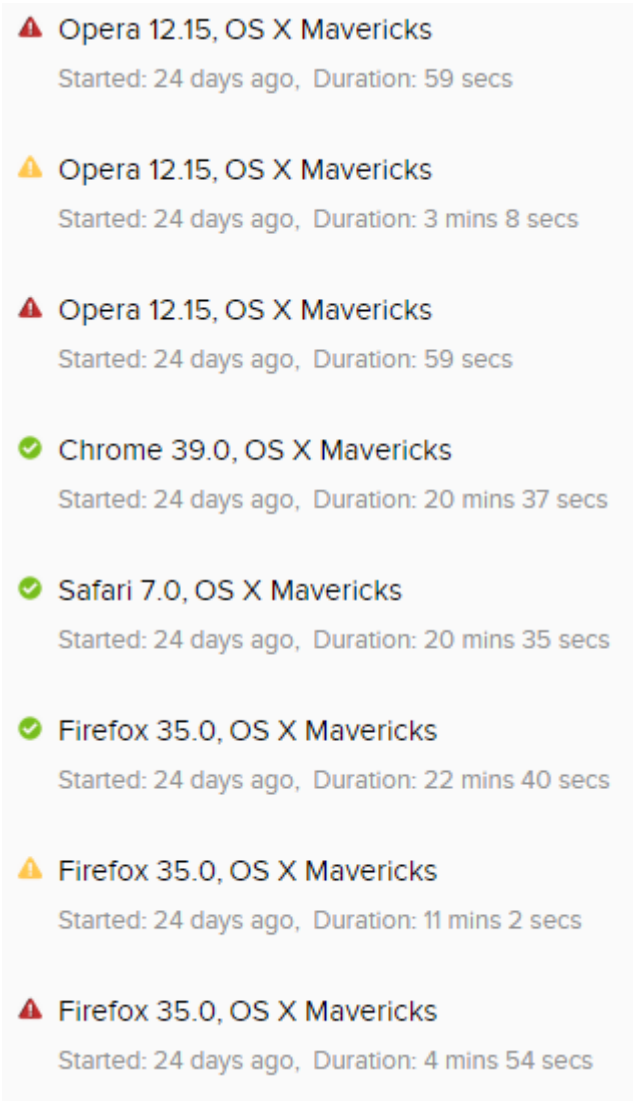


Figura 2.14 – Pruebas Browserstack

A continuación se realizan las pruebas con el sistema operativo Mavericks de Apple, experimentando algunos inconvenientes con el navegador Firefox y luego con el sistema Opera.

Mientras que con Firefox los problemas se solucionaron realizando nuevamente las pruebas, para el caso de Opera e IE no se pudieron obtener valores de pruebas.

Finalmente, algunas pruebas tuvieron que ser repetidas para corroborar los valores obtenidos en las primeras pruebas

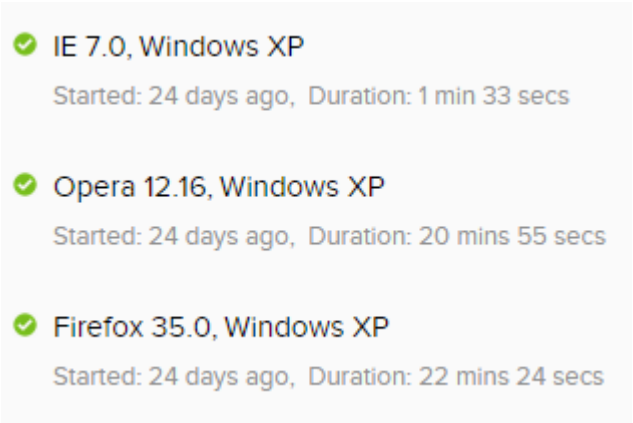


Figura 2.15 – Pruebas Browserstack

2.2.4 Resultados

Luego de realizadas las pruebas se toman los valores de "*median_rtt*" (med) y "*dev_rtt*" (sigma) de cada uno de los escenarios, pudiendo generar así la Tabla 2.6:

OS/Browser	Windows XP		Windows 7		Windows 8.1		Mac OS	
	med	sigma	med	sigma	med	sigma	med	Sigma
Chrome	19	2	23	2	25	5	14	1
Firefox	109	44	18	3	19	4	19	2
Safari	No se pudo realizar la prueba por falta de drivers		20	3	24	5	8	0
IE	El Javascript no funciona correctamente. No comienza la prueba		20	3	20	3	No esta soportado	
Opera	16	2	18	2	22	6	Error al intentar conectarse al server local	

Tabla 2.6 – Resultados pruebas Browserstack

A su vez se construye una tabla en la base de datos del proyecto con las siguientes características:

id	os	browser	median_rtt	dev_rtt	norm_factor
1	Win7	chrome	23	2	0
2	Win7	firefox	18	3	5
3	Win7	safari	20	3	3
4	Win7	IE	20	3	3
5	Win7	opera	18	2	5
6	WinXP	chrome	19	2	4
7	WinXP	firefox	109	44	-86
8	WinXP	opera	16	2	7
9	Win8.1	chrome	25	5	-2
10	Win8.1	firefox	19	4	4
11	Win8.1	safari	24	5	-1
12	Win8.1	IE	20	3	3
13	Win8.1	opera	22	6	1
14	MacOS	chrome	14	1	9
15	MacOS	firefox	19	2	4
16	MacOS	safari	8	0	15

Tabla 2.7 – Tabla en DB de Simón

Como se puede observar entre las Figuras 2.10 y 2.14, existieron escenarios de pruebas, que no pudieron ser llevados a cabo. En particular para el caso de *Windows XP* y las pruebas con *Safari*, los problemas están asociados a la falta de un software en *Windows XP* necesario para ejecutar el navegador de manera correcta. Dado que la máquina virtual utilizada, es parte de la plataforma de *Browsersack* la instalación de este tipo de software no puede ser realizado en modo *Automate.* Para el caso de *IE*, el script del medidor no se ejecutaba correctamente, por lo que no se generaba ninguna medida. Se espera por lo tanto que no se obtengan medidas con esta configuración.

En lo que respecta a las pruebas con *MacOS, IE* no está soportado en las plataformas *Apple*.
Por el lado del navegador *Opera*, se obtienen los siguientes *logs* visuales, portados por la herramienta *Browserstack*:

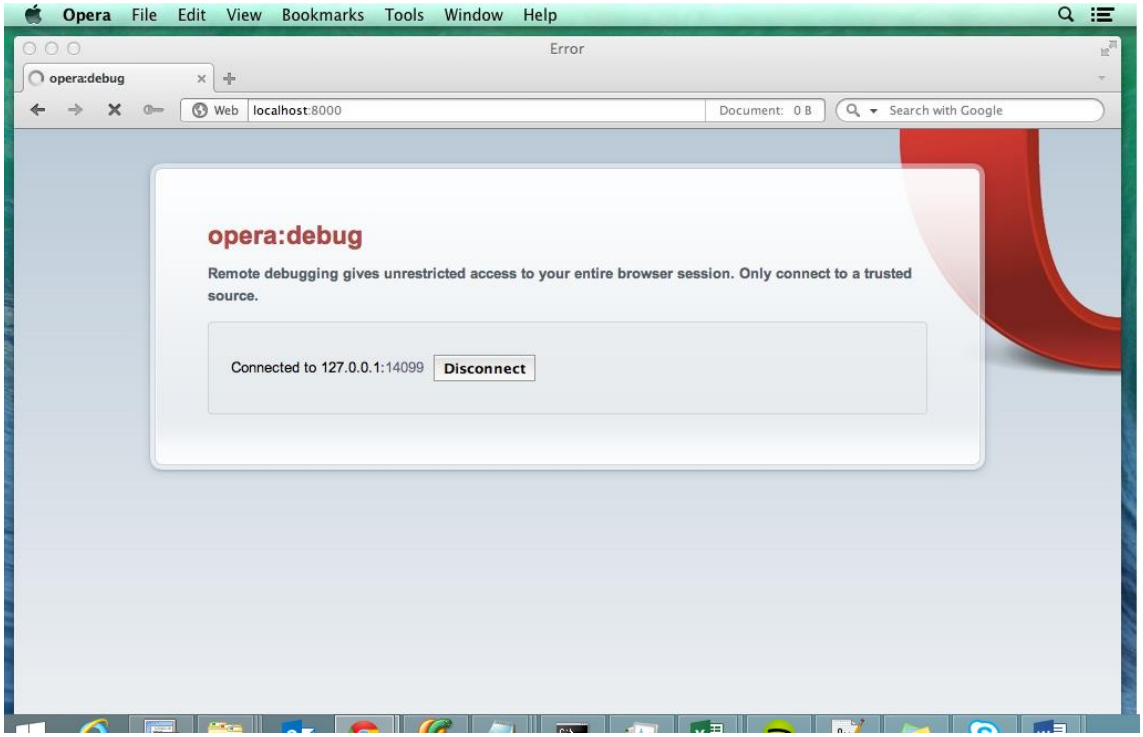


Figura 2.16 – Logs visual pruebas Apple/Opera

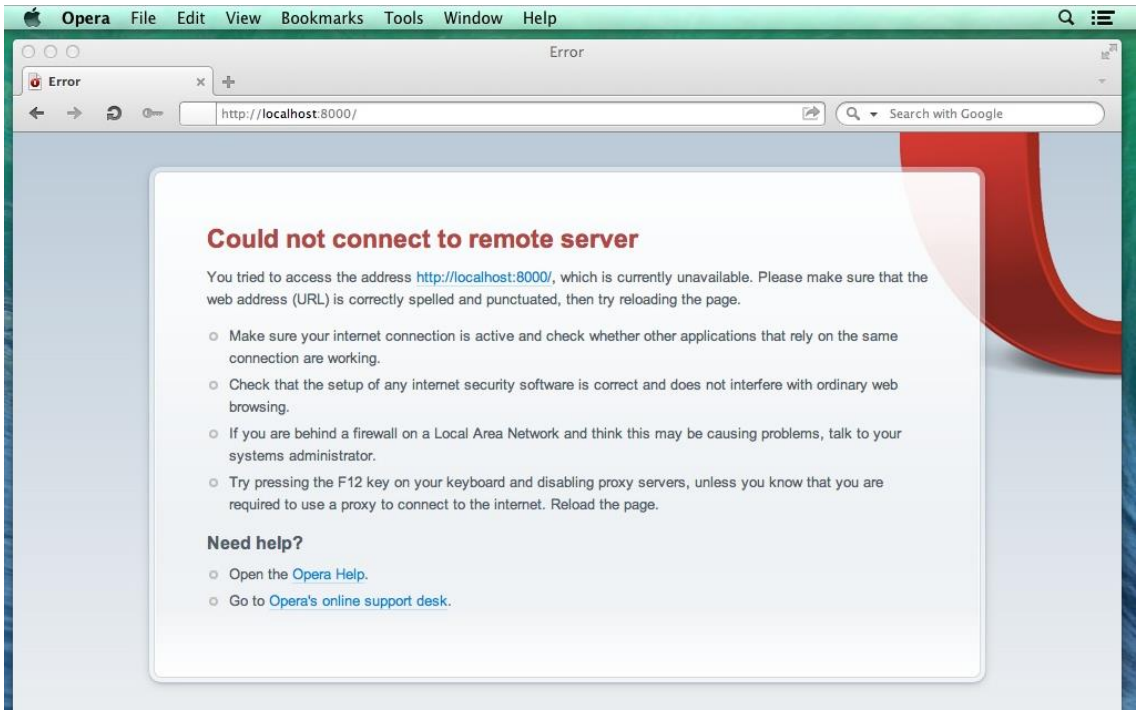


Figura 2.17 – Logs visual pruebas Apple/Opera

Los errores mencionan la dificultad para conectarse a la página de pruebas con la configuración *MacOs – Opera*. En particular para este caso aún se están revisando las razones por las cuales no ha funcionado, pero por el momento y teniendo en cuenta el alcance de este proyecto, se considera que las pruebas no tuvieron el resultado esperado.

Teniendo en cuenta el objetivo de esta etapa del proyecto, se construye la tabla con los valores de normalización necesarios, teniendo como dupla base *Windows 7 y Chrome.*
El Factor de corrección de obtiene realizando la resta entre la mediana de la dupla base con la mediana de la dupla a normalizar.

OS/Browser	Windows XP		Windows 7		Windows 8.1		Mac OS	
	Factor de corrección	sigma	Factor de corrección	sigma	Factor de corrección	sigma	Factor de corrección	sigma
Chrome	4	2	0	2	-2	5	9	1
Firefox	-86	44	5	3	4	4	4	2
Safari		0	3	3	-1	5	15	0
IE		0	3	3	3	3		0
Opera	7	2	5	2	1	6		0

Tabla 2.8 – Tabla de factor de normalización

Es importante recalcar que la creación de la tabla en la base de datos permite realizar normalizaciones con cualquier dupla existente.

Capítulo 3 - Incorporación de mediciones de partners internacionales

3.1 Introducción general

Además de los medidores ya existentes en el proyecto Simón, es importante incorporar más fuentes de datos al proyecto, tanto dentro como fuera de la región.

Las medidas de latencia se han llevado a cabo por diversas entidades desde los comienzos de Internet, dada la importancia de las mismas para el correcto desarrollo de aplicaciones y servicios sobre la red. Esto hace que se puedan investigar fuentes alternativas de datos, de manera de enriquecer la base de datos de Simón con resultados independientes y con enfoques de medición distintos.
Para esta etapa se planean incorporar mediciones del proyecto ATLAS de RIPE [40] y Archipiélago de CAIDA [41].

La incorporación de estas mediciones conforma uno de los entregables más importantes de esta contribución, ya que bases de datos de estas entidades son muy ricas. Al incorporar mediciones de estos proyectos se estaría contando con más medidores, más puntos de medición, y más mediciones de forma casi gratuita.
A su vez la incorporación de estas medidas permite la manipulación, comparación y análisis de los mismos con las herramientas de trabajo y la arquitectura desarrolladas dentro del proyecto Simón, teniendo contenida en una sola base de datos, mediciones de 3 fuentes distintas.

3.2 RIPE NCC

El *Réseaux IP Européens Network Coordination Centre* (RIPE NCC) es el registro de direcciones de Internet correspondiente a la región de Europa y parte de Eurasia, de la misma manera que LACNIC [42] lo es para la región de América Latina y el Caribe.

Desde su creación el rol más importante de los Registros Regionales de Internet (RIR por sus siglas en inglés) fue velar por el registro de las direcciones IP públicas, permitiendo así llevar un control de quien hace uso de los recursos y quien es responsable por los mismos. Además de ese rol principal y como forma de brindar a la comunidad servicios de valor agregado, RIPE NCC ha desarrollado un proyecto llamado RIPE Atlas [43], una red global de *probes* (sondas de medición) que obtienen datos de conectividad y accesibilidad, permitiendo medir el estado de Internet en tiempo real.
En la Figura 3.1 podemos apreciar la última versión de *probe* utilizada por RIPE:

Figura 3.1 – Probe de RIPE atlas [42]

A modo de comprender el funcionamiento solicitamos una de estas pruebas a RIPE, las cuales son distribuidas de manera gratuita. Solamente debe ser detallado la razón del pedido y una dirección e contacto y envío fidedigna.
Se pueden destacar algunas características interesantes del equipo:

- Es pequeño, de 7x7x2 centímetros
- No requiere configuración, es *plug & play*
- Basado en sistemas *Opensource*

La *probe* se conecta directamente a un *router* hogareño mediante un cable UTP, sin necesidad de realizar ninguna configuración adicional.
Luego de conectado, debe ser registrado en la página web de RIPE y puede ser luego controlado a través de la misma página.

Se puede ver un detalle de parte de la consola de administración del equipo en la Figura 3.2:

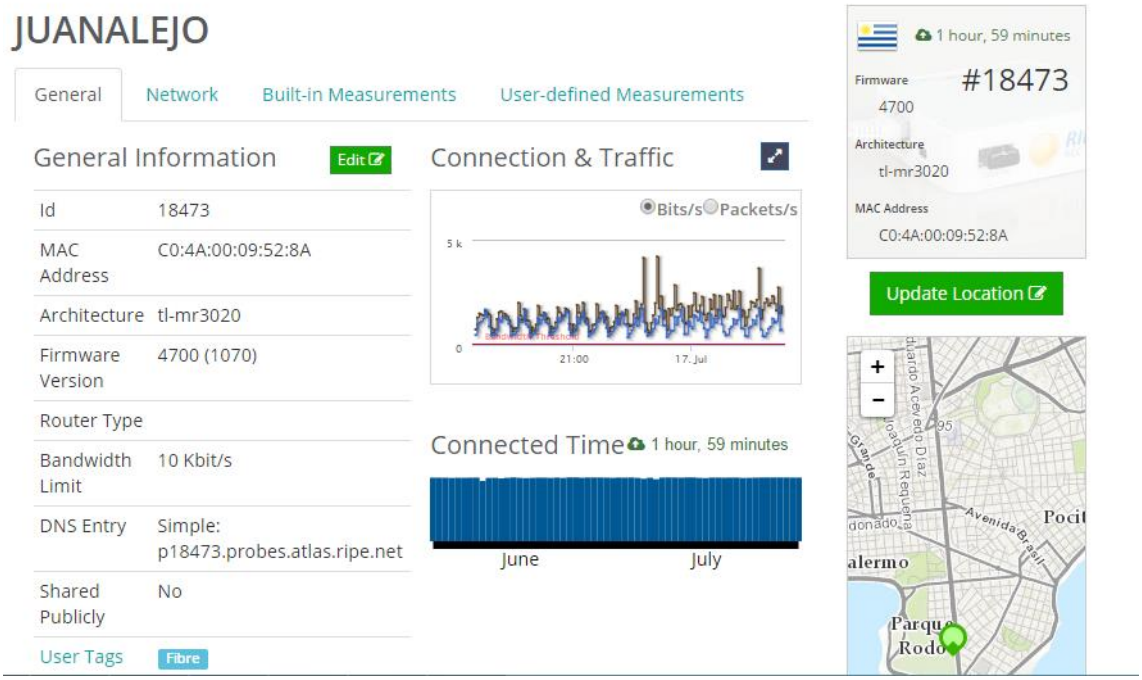


Figura 3.2 - Consola de Administración de probe

Mediante la *probe* obtenida, se busca entender el formato de medidas que se obtienen a través de las mismas, para luego programar scripts que obtengan la información útil para el proyecto.

Para tener en cuenta cuan extensa y útil puede llegar a ser la información obtenida por las *probes* podemos observar una aproximación de la cobertura, en cantidad de *probes*, reflejada en la Figura 3.3:

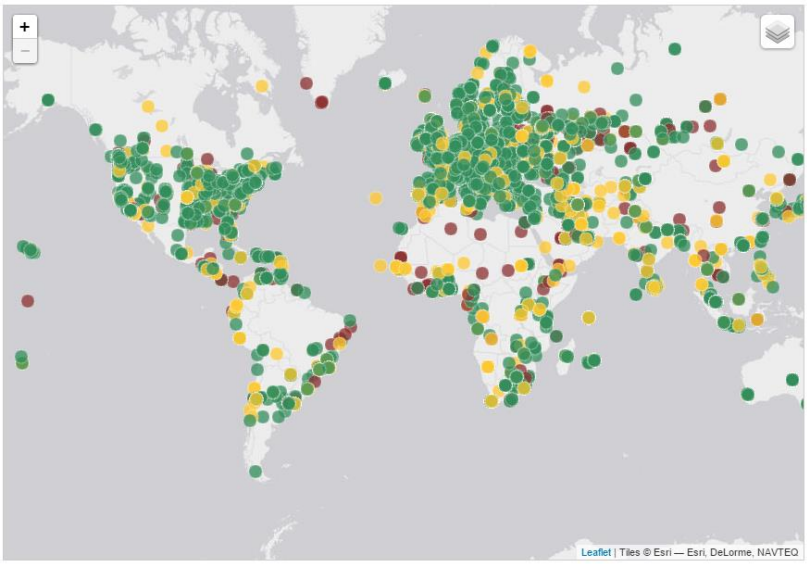


Figura 3.3 – Mapa de distribución de Probes a nivel mundial [44]

Siendo más específicos a la región de las Américas:

Figura 3.3 – Mapa de distribución de Probes a nivel de América Latina [45]

Los puntos verdes corresponden a *probes* conectadas, los puntos amarillos a *probes* desconectadas y los rojos a las que se encuentran abandonadas.
En total, a nivel mundial se pueden encontrar más de 8.300 *probes* conectadas, de las cuales algo más de 150 se encuentran en la región de LACNIC.

Existe otro elemento más de medición llamado *anchor* (ancla en inglés), el cual tiene un funcionamiento muy similar al de una *probe*, pero con más capacidad.
En general los Achors están dispuestos en puntos con capacidad de conectividad lo suficientemente alta como para soportar grandes cantidades de consultas. Es por esta razón que en general están soportados bajo redes de Proveedores de Internet (ISPs) u organizaciones vinculadas a Internet (NIC.br, ICANN, LACNIC, ANTEL, etc.)

Podemos encontrar 134 *Anchors* a nivel mundial:

Total number of RIPE Atlas anchors: 134

Figura 3.4 – Mapa de distribución de Anchors a nivel mundial [46]

Para generar una medida es necesario hacerlo de manera manual, eligiendo el destino (una *probe o anchor*) también de manera manual, dentro de la consola web de administración de la *probe.* Luego se elige que tipo de prueba será iniciada y la duración que se considera debe tener la prueba realizada.
A efectos del estudio realizado en esta sección, tanto las *probes* como los *Anchors* serán considerados de manera igualitaria como destinos y orígenes de mediciones. Esto significa que los resultados obtenidos, serán origen-destino: *probe-probe*, *probe-anchor o anchor-probe.*

Las *probes* pueden generan medidas de distintos tipos entre sí:

1. Ping ICMP
2. Traceroute
3. DNS
4. SSL

Estas medidas son generadas por cada usuario, poseedor de una *probe* y luego guardadas en las bases de datos de RIPE NCC.
En esta sección nos concentraremos únicamente en la funcionalidad de *ping* mediante el protocolo ICMP estándar.

RIPE NCC llama "comunidad" [47] al conjunto de participantes de esta red, clasificados de la siguiente manera:

a) *Host*: Es cualquier persona que conecte una *probe* o un Anchor a su red. Cada host puede generar o consultar medidas, en base a un sistema de créditos ganados y utilizados.
b) *Sponsor*: Es un individuo u organización que apoya financieramente el proyecto Atlas.
c) *Embajador*: Es un individuo u organización que distribuye las *probes*.

Es importante destacar que a pesar de enfocar los esfuerzos de este proyecto en la región de América Latina y el Caribe, en particular en las medidas de *ping,* es posible extender el ámbito de acción, utilizando la información provista con destinos por fuera de la región de influencia de LACNIC.

.

3.3 Obtención de datos de RIPE Atlas

3.3.1 Introducción

La base de datos contenida en los servidores RIPE Atlas es tremendamente grande y compleja. Es por esto que se procede a realizar una obtención de datos en etapas, desagregando así las complejidades de la base de datos consultada. Cada una de las etapas permite realizar un manejo local de la información, evitando la necesidad de estar continuamente realizando consultas web, pudiendo así trabajar *offline* con datos persistidos de manera local.
La opción de trabajar en etapas fue definida luego de realizar una maqueta inicial con todas las funcionalidades juntas en un solo script. Esta opción a pesar de tener mejor performance y ocupar menos memoria (no se utilizan archivos de texto para guardar información) tiene ciertas complejidades que hacen más difícil de manejar los datos. Como ejemplos podemos mencionar:

- En caso de no trabajar en etapas, la conexión a internet debe ser muy estable, ya que cualquier pérdida de conectividad hace fallar todo el proceso.
- El script se complejiza, haciendo más difícil la detección de errores.
- Al momento de integrar al proyecto Simón, la desagregación en etapas permite decidir a qué modulo pertenece cada funcionalidad.
- Un único script, hace vulnerable el proceso desde el punto de vista de seguridad. Al manejar el proceso en etapas, solo parte de las funcionalidades tienen la necesidad de realizar consultas web ("hacia afuera"), siendo vulnerable solo parte del proceso.

La obtención de los datos se hace a través de scripts programados en lenguaje *Python* y consultas del tipo REST.

En esta sección utilizaremos las funcionalidades detalladas en la "RIPE Atlas REST Api" [48] para acceder a los datos contenidos en la base de datos de RIPE NCC.

La base debe ser consultada en línea, esto quiere decir que es necesario acceder a ciertas urls con determinados filtros, obtenidos a través de la guía de APIs, obteniendo como resultado una respuesta HTTP, en la cual el contenido del cuerpo del mensaje esta codificado en formato JSON.

La descripción general del proceso se puede observar en la Figura 3.5:

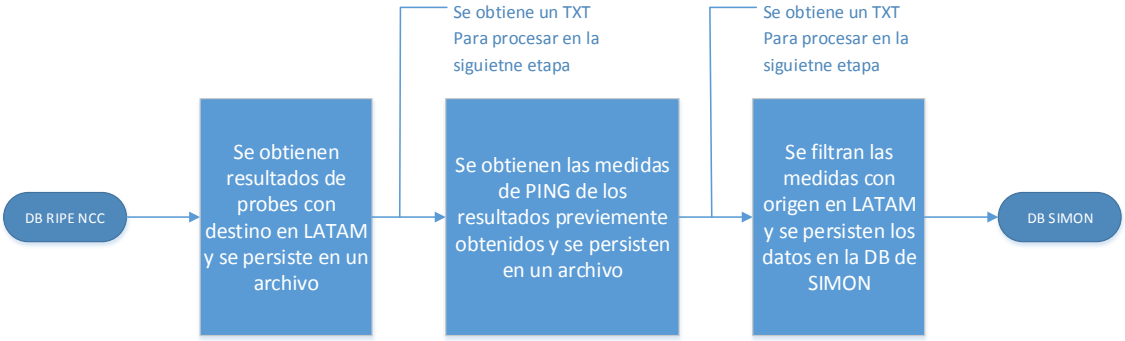


Figura 3.5 – Diagrama de etapas

<u>Nota Importante</u>
Las funcionalidades de la API fueron modificadas por RIPE NCC durante el año 2015. Dado que las pruebas realizadas en esta sección fueron mayormente llevadas a cabo a finales del año 2014 y principios del año 2015, el script correspondiente a la Etapa 1 tuvo que ser modificado de manera de poder ser integrado al ambiente de producción de Simón.
Esta integración aún se está llevando a cabo al día de entrega de este documento, por lo que se tendrán en cuenta las pruebas realizadas SIN las modificaciones, realizadas antes del cambio de las APIs, ya que el nuevo script de la Etapa 1 aún no fue probado.
Es importante destacar que las Etapas 2 y 3 no se ven afectadas por la modificación de la API.

En la siguiente sección se detalla el comportamiento de los scripts realizados en cada etapa. Varias de las librerías de *Python* utilizadas en lo scripts, se repiten en todas las etapas, por lo que no serán detalladas en las explicaciones. Las librerías son:

- Json – permite cargar diccionarios del tipo JSON para ser manipulados.
- Urllib – permite realizar consultas web y cargar el contenido de las respuestas HTTP.

3.3.2 Etapas definidas para obtención de datos

ETAPA 1 [49]

En esta esta etapa se procede a buscar todos los elementos de la base de datos de RIPE NCC que contengan como destino, una IP correspondiente a las asignadas a LACNIC por la IANA para ser utilizadas en la región de América Latina y el Caribe.

Para realizar esto se utiliza como base los siguientes bloques /8, todos asignados a LACNIC:

bloques = ['177/8', '179/8', '181/8', '186/8', '187/8', '189/8', '190/8', '191/8', '200/8', '201/8']

Como fue mencionado, la forma de obtener información de la base de datos de RIPE es a través de una consulta HTTP, por lo tanto se utiliza la siguiente url como base:

url = https://atlas.ripe.net/api/v1/measurement/?dst_addr=bloques[i]&limit=100

donde "bloques[i]" corresponde a cada uno de los bloques contenidos en el conjunto anteriormente descrito.

Los parámetros de la API utilizados en la url son:

- *dst_addr* - correspondiente a los bloques de destino de las medidas que deseamos obtener
- *limit* - establece la cantidad de resultados que deseamos obtener cada vez que se haga la consulta.

Es importante destacar que el límite máximo de objetos por consulta es 100, que no es necesariamente el total de resultados, pero la base de datos de RIPE NCC funciona con un sistema de "paginado", para evitar sobrecargar cada una de las consultas.
Es por esta razón que es necesario recorrer todas las páginas para cada uno de los bloques que deseamos realizar la consulta.

Antes de proceder a explicar el mecanismo de recorrido página a página y la obtención de los datos de cada una, se debe tener en cuenta como se estructura la información que brinda cada respuesta HTTP de la base de datos de RIPE.

Las respuestas obtenidas al consultar por la url mencionada están compuestas de 2 objetos principales en formato JSON:

- *meta* – donde se detalla la información general de la consulta realizada
- *objects* – donde se listan todos los resultados que concuerdan con los filtros establecidos en la consulta.

El primer chequeo realizado en el script, se hace sobre el objeto "meta", compuesto de datos que nos permiten controlar el paginado de las consultas.

A continuación se detalla el objeto mencionado:

{"meta":{"total_count":1049925,"use_iso_time":false,"next":"/api/v1/measurement/?limit=100&offset=100","limit":100,"offset":0,"previous":null}

Resaltado en amarillo se observan los dos campos claves: *next* y *offset*.

El primer campo detalla cómo sería la url de la siguiente página. Esto quiere decir que si completáramos la url [http://atlas.ripe.net] con el campo *next*, la consulta devuelve la siguiente página, con los mismos filtros de la consulta original. Se puede observar que ese campo está directamente relacionado con el segundo mencionado, *offset*. Ese campo es el encargado de realizar el corrimiento de los resultados, respecto al primer resultado guardado en la base de datos. Por ejemplo, si deseamos obtener 100 resultados, pero de algún lugar intermedio de la lista, se detalla el valor *offset* =x, dentro de la url de consulta.
Dentro de la API, existen otros campos que pueden ser de utilidad, pero que no se tienen en cuenta para el script de esta sección.

A nivel de programación, se identifica como un bucle "for" al recorrido de los bloques /8 y al paginado como un bucle de la función "while", ya que se realiza el paginado hasta obtener NULL en el campo *next,* lo que permite identificar la última página.

Como fue mencionado, medida que se realizan las consultas, los resultados correspondientes al filtro aplicado son guardados en un archivo de texto llamado "ripe_db.txt". El contenido del archivo es texto plano, con formato JSON. De esta forma obtenemos un conjunto de datos estable sin la necesidad de consultar en línea de manera recurrente.

Cada uno de los resultados obtenidos en esta etapa posee un formato particular. A modo de ejemplo representamos un resultado a continuación, donde se resaltan los ítems más importantes a tener en cuenta:

{"status": {"id": 4, "name": "Stopped"}, "creation_time": 1395873042, "participant_count": 116, "dst_addr": "190.186.210.16", "type": {"id": 1, "name": "ping", "af": 4}, "dst_asn": 25620, "all_scheduling_requests_fulfilled": true, "resolve_on_*probe*": false, "is_oneoff": true, "packets": 3, "af": 4, "dst_name": "190.186.210.16", "is_public": true, "resolved_ips": ["190.186.210.16"], "stop_time": 1395873604, "can_visualise": false, "interval": null, "result": "/api/v1/measurement/1601368/result/", "msm_id": 1601368, "description": ""}

Los pares atributo-valor resaltados representan:

name – Corresponde al tipo de prueba realizada. Como mencionamos anteriormente la misma puede ser ping, *traceroute, dns o ssl*.
dst_asn – Corresponde al Numero de Sistema Autónomo (ASN por sus siglas en inglés) de destino

result – Corresponde a la url donde se puede obtener los valores obtenidos en el resultado, en este caso de un PING

Estos campos son importantes para la siguiente etapa, donde se tendrán en cuenta para generar los filtros de la API y los bucles en el script correspondiente.

ETAPA 2 [50]

En esta etapa se utiliza el archivo de texto obtenido en la primera etapa "ripe_db.txt", teniendo en cuenta los pares atributo-valor resaltados en la etapa anterior.

Al igual que en la etapa anterior, se debe realizar una nueva consulta a la base de datos en línea, en este caso utilizando como url base la obtenida en el atributo-valor de *result*, teniendo en cuenta el atributo-valor *name*, para consultar solamente por los resultados que corresponden a pruebas PING.

Parte del pseudocódigo utilizado en el script es el siguiente:

```
WHILE hay_lineas_en archivo:
        IF medida['type']['name'] == 'ping':
                resultado = abrir_url("https://atlas.ripe.net/medida['result']")
```

El resultado obtenido de la consulta HTTP en este caso es también del tipo JSON.
La información obtenida en la consulta contiene lo que se desea guardar en la base de datos de Simón:

- tiempo de latencia, máximo, mínimo, media
- origen y destino
- fecha de la prueba

Antes de proceder a guardar la información en otro archivo de tipo texto se genera un resultado JSON personalizado conteniendo también el valor de "dst_asn" correspondiente al número de sistema autónomo del destino, que por defecto no viene incluido en el resultado obtenido de la consulta al campo *result,* sino que hay que obtenerlo del archivo de texto procesado en la etapa 1.

Por lo tanto la información guardada posee el siguiente formato:

{"info": {"asn_dst": 22047}, "resultado": [{"min": 218.02, "from": "190.103.184.148", "size": 12, "fw": 4580, "af": 4, "dup": 0, "step": null, "dst_name": "190.45.0.20", "rcvd": 3, "result": [{"rtt": 218.654}, {"rtt": 218.02}, {"rtt": 218.149}], "msm_id": 1439796, "dst_addr": "190.45.0.20", "sent": 3, "proto": "ICMP", "avg": 218.27433333333332, "lts": 179, "max": 218.654, "prb_id": 10191, "group_id": 1439796, "msm_name": "Ping", "type": "ping", "timestamp": 1392396249},{SIGUIENTES RESULTADOS}]}

El archivo obtenido luego de realizar el proceso en esta etapa, se denomina "results_ripe.txt", que es finalmente el archivo que debemos procesar para guardar las medidas obtenidas en la base de datos.

ETAPA 3 [51]

En esta etapa se realiza un filtro final y se persisten los resultados obtenidos en la base de datos de Simón.

Teniendo en cuenta la información contenida en el archivo de texto creado en la etapa anterior, obtenemos los valores requeridos de la prueba PING ICMP realizada con la infraestructura de RIPE:

- RTT mínimo
- RTT máximo
- RTT promedio,
- Fecha de la prueba
- ASN destino
- IP Origen
- Paquetes enviados
- Otros valores que no serán tomados en cuenta por ahora en este análisis.

A pesar de ser generados por otra herramienta, se guardarán los datos de RIPE, en las columnas correspondientes a sus mismos nombres en la base de datos de Simón.

A su vez en esta etapa se realiza el control de origen del resultado obtenido, ya que se buscan guardar en la base de datos de Simón, medidas dentro de la región de América Latina y el Caribe.

Luego de realizar los filtros necesarios, se pasa a la persistencia de los resultados, en la cual hay que tener algunas consideraciones al ingresar los datos.
Utilizaremos las funcionalidades que brinda la plataforma *Django* para acceder a la base de datos, sin enviar código SQL directamente desde el script, en particular la API *models* como fue mencionado al inicio de este documento.
A su vez recurrimos a dos clases creadas en el proyecto Simón: *Results* y *AS.* Estas clases hacen uno de la API y de los métodos que tiene configurada, en particular el método *save()*, que guarda los campos establecidos en la clase *Results* en la base de datos.

Los campos establecidos por Simón que componen la tabla de resultados son:
date_test, versión, ip_origin, ip_destination, testype, number_probes, min_rtt, max_rtt, ave_rtt, dev_rtt, median_rtt, packet_loss, country_origin, country_destination, ip_version, tester, tester_version, as_origin_id, as_destination_id, user_agent, url

Los campos obligatorios en la base son los siguientes:
date_test, version, ip_origin, ip_destination, testype, country_origin, country_destination, ip_version, tester, tester_version.

Por lo tanto, aunque no obtengamos información relevante para esos campos de los resultados de RIPE, todos deben ser completados. Esto genera un problema en lo que respecta a los países de origen (*country_origin)* y destino (*country_destination)*, ya que esta información no viene por defecto en los resultados de RIPE.
Para adquirir esta información se manejan dos opciones dentro del script. La primera implica la utilización de una función previamente desarrollada en el proyecto Simón, que devuelve el código de país en formato ISO, en función de la IP que se le ingresa. Esta función no se encuentra desarrollada en su totalidad, por lo que se optó por un camino alternativo. Esta segunda opción, que finalmente fue la utilizada, es consultar nuevamente la base de datos de RIPE NCC, con la diferencia que en este caso se consulta a otra sección web:

https://stat.ripe.net/data/geoloc/data.json?resource=x.x.x.x

Donde x.x.x.x corresponde a la dirección IP de la cual queremos obtener la información.
La respuesta a esta consulta es nuevamente de formato JSON, con un campo donde se detalla el país, donde se encuentra geo-localizado el bloque que contiene la IP en cuestión.
Un ejemplo de resultado de la consulta es el siguiente:


```
{
    "status": "ok",
    "server_id": "stat-app6",
    "status_code": 200,
    "version": "2.2",
    "cached": false,
    "see_also": [],
    "time": "2015-07-20T03:06:02.168095",
    "messages": [],
    "data_call_status": "supported - connecting to ursa for all data
sets",
    "process_time": 182,
    "query_id": "40aa2008-2e8c-11e5-8a9c-549f3540cd49",
    "data": {
        "query_time": "2015-07-07T00:00:00",
        "resource": "177.43.115.148",
        "locations": [
            {
                "city": "",
                "prefixes": [
                    "177.43.114.0/23"
                ],
                "country": "BR",
                "longitude": -46.6358,
                "covered_percentage": 100.0,
                "latitude": -23.5477
            }
        ],
        "unknown_percentage": 0.0
    }
}
```


Se observa resaltado en color amarillo, la sección que se desea obtener de esta consulta. Es importante mencionar que la utilización de esta opción es fácilmente

modificable dentro del código del script. En futuras actualizaciones del proyecto se podrá consultar directamente una base interna, que permitirá ahorrar tiempo y mejorar la performance de los procesos de persistencia en la base de datos.

Teniendo en consideración lo mencionado anteriormente, tomando solo en cuenta el pasaje a la base de datos, la persistencia de un resultado se resume con el siguiente código:

```
results = Results() #Instancia de la Base de Datos
results.tester = testype
results.ip_origin = ip_origin
results.ip_destination = ip_destination
results.as_destination = AS.objects.get_as_by_ip(results.ip_destination)
results.as_origin = AS.objects.get_as_by_ip(results.ip_origin)
results.country_destination = country_destination
results.country_origin = country_origin
results.set_date_time(str(date_test.date()), str(date_test.time()))
results.version = ip_version
results.ip_version = ip_version
results.min_rtt = min_rtt
results.ave_rtt = ave_rtt
results.max_rtt = max_rtt
results.packet_loss = packet_loss
results.save() #Se persiste en la DB
```

Dando entonces por finalizado, el proceso total de obtención y almacenamiento de un resultado de la base de datos de RIPE NCC

3.4 CAIDA

CAIDA, *Center for Applied Internet Data Analysis*, es un compromiso de colaboración entre organizaciones comerciales, gobiernos y sectores de investigación, que desde el año 2007 se encuentran encaminados en promover una mayor cooperación en la ingeniería y el mantenimiento de Internet como un sistema robusto, escalable y global.

Esta organización, de manera similar a RIPE NCC, ha desarrollado una infraestructura de mediciones globales, basadas también en la colocación de generadores (tomadores) de medidas. Estos equipos son llamados *Monitors*, basados en los modelos de *Rasperry Pie,* se encuentran desplegados en la mayoría de los casos dentro de redes robustas, como pueden ser Proveedores de Internet u Organizaciones similares a CAIDA (Grandes Universidades, RIRs, IXPs, etc).

Figura 3.6 – CAIDA monitor [53]

A continuación se muestran las regiones con mayor densidad de *Monitors*.

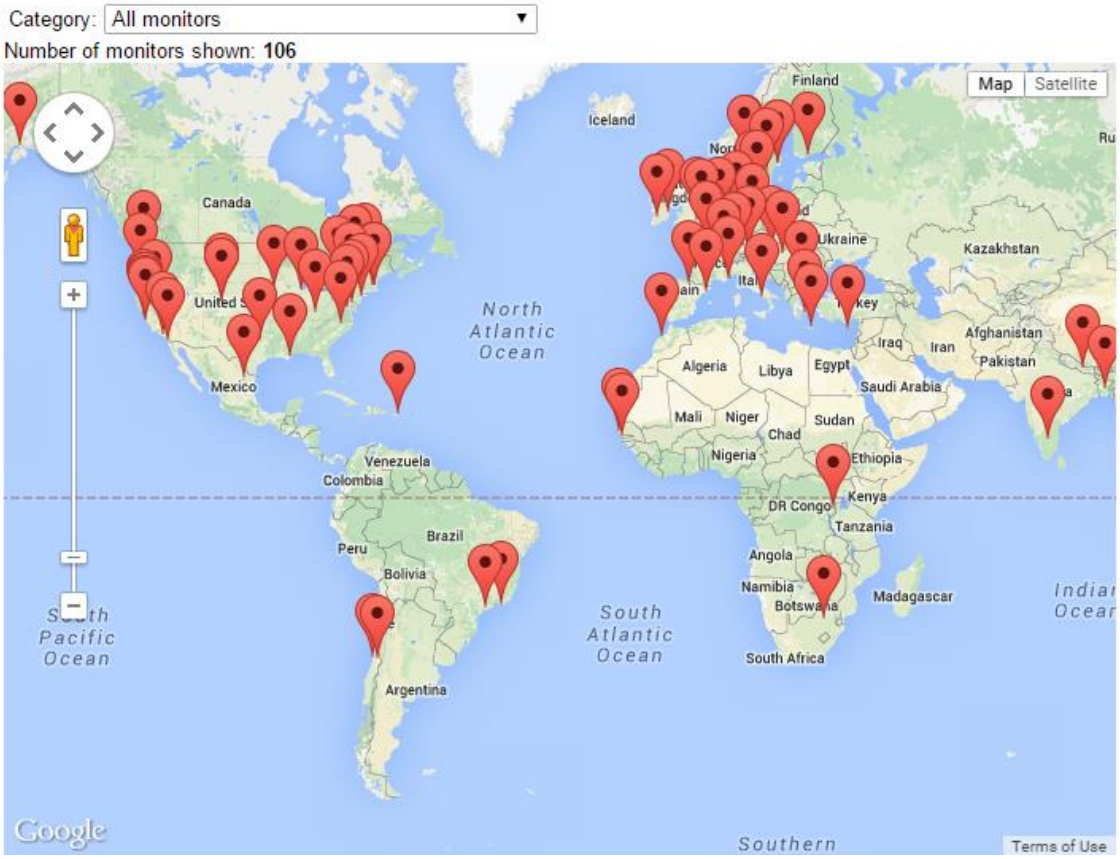


Figura 3.7 – Despliegue de medidores proyecto Archipiélago Ark [54]

Del total de monitores a nivel mundial, 119, solamente 4 se encuentran en la región de América Latina y el Caribe. Dos de ellos en Chile, dos en Brasil y uno en Puerto Rico. Dado que este último no se encuentra bajo la cobertura de LACNIC, no será tenido en cuenta para este proyecto.
A pesar de ser un número bajo dentro de la región, la cantidad de datos generados por los mismos es extremadamente grande

CAIDA posee un abanico variado de proyectos, pero en esta etapa el foco está centrado en el proyecto "*Archipielago (Ark*)" [52], que apunta a reducir los esfuerzos necesarios para desarrollar e implementar medidas sofisticadas a gran escala.
En importante resaltar que los datos públicos a los cuales es posible acceder corresponden a mediciones previas al 2013 inclusive.
LACNIC se encuentra en negociaciones para que la información más reciente sea facilitada.

En este caso los *Monitors* también pueden generar varios tipos de pruebas:

- Ping
- Traceroute
- Medidas TCP, UDP
- Paris-traceroute

Los mismos utilizan una herramienta llamada *Scamper*, que permite realizar medidas de manera flexible utilizando los tipos de prueba antes mencionado, en particular utilizan Paris-traceroute, un tipo modificado de *traceroute,* para tomar las medidas. En la siguiente sección se desarrollaran las ideas aquí descritas.

3.5 Obtención de datos Archipielago (Ark) - CAIDA

3.5.1 Introducción

Previo a explicar la metodología utilizada en esta contribución para la obtención de datos para enriquecer las medidas de Simón, es necesario explicar la metodología utilizada por CAIDA para obtener sus resultados de Internet.

Inicialmente se realiza la elección de destino. Se elige un destino IP de manera aleatoria, de cada bloque /24 publicado en internet, siendo alcanzada al menos una vez, en un ciclo de prueba predefinido por CAIDA de 48 horas, llamado *probing cycle.* La lista de prefijos utilizados para realizar las pruebas es generada utilizando *RouteViews* [55].
A su vez se utiliza la metodología de trabajo *team-probing* por parte de los medidores. Esta metodología define conjuntos de medidores, de manera que cada destino elegido sea testeado por un monitor por cada ciclo de prueba [56].

Otro punto importante de la metodología utilizada por los medidores es que se utiliza el tipo de *traceroute* conocido como *Paris-traceroute* [57]. Este *traceroute* tiene la particularidad que durante un ciclo de prueba, se mantiene el camino de todos los paquetes de prueba enviados, minimizando en gran medida los errores introducidos en las medidas por el balanceo de carga realizado por los *routers* intermedios entre el origen (medidor) y el destino (IP aleatoria). En particular el estudio de este tipo de *traceroute* se centra en 3 tipos de balanceo de carga:

- *Per-flow load balancing*: se identifica un flujo de datos a través de los datos contenidos en la cabecera de los paquetes IP, como pueden ser IP destino, IP origen, protocolo, puerto origen y puerto destino entre otros. Esto hace que si alguno de estos factores cambia, por ejemplo los puertos de origen o destino, el flujo de datos puede cambiar, de manera de no seguir la misma ruta, a pesar de mantenerse el origen y destino.
- *Per-packet load balancing:* se basa solamente en capacidades de las interfaces del router o balanceador. Este tipo de balanceo lo hace de manera de mantener las cargas balanceadas.
- *Per-destination load balancing:* este tipo de balanceo tiene solo en cuenta el destino, por lo que no define ningún tipo de flujo en el camino.

La idea principal *Paris- traceroute* es mejorar la performance del *traceroute* por defecto, frente a la presencia de balanceo de carga por flujo, manipulando los valores de:

- Para pruebas TCP se varía el campo de numero de secuencia
- Para pruebas UDP se varía el campo de *checksum.*

Las características de este *traceroute,* definen de qué manera los datos aportados por CAIDA van a ser almacenados en la base de datos de Simón.

Aprovechando la gran cantidad de información de latencias obtenidas entre cada uno de los saltos del *traceroute* y no solamente entre origen y destino, se decide almacenar también la latencia contenida entre los saltos pertenecientes a la región de estudio. Se asume que la resta de las latencias obtenidas en durante el camino son fiables en caso de que se midiera los puntos de manera aislada. Por ejemplo, si desde origen al punto 4 hay XX milisegundos de latencia y desde origen al punto 5 hay YY milisegundos de latencia, desde 4 a 5 hay (XX-YY) milisegundos de latencia.
Se toma en cuenta que esta decisión puede introducir errores, ya que las medidas entre los puntos intermedios no fueron realizadas, sino que calculadas. A pesar de esto, se asume que la gran cantidad de datos aportados, diluyen los posibles errores, siendo por lo tanto las latencias medias calculadas entre dos puntos, confiables dentro de los alcances de este proyecto.
Para apoyar esta decisión se recurre a algunas estadísticas generadas por CAIDA, del medidor "scl-cl" donde se puede encontrar la siguiente gráfica en la cual se comparan la cantidad de *traceroutes* realizados en función de los saltos IP:

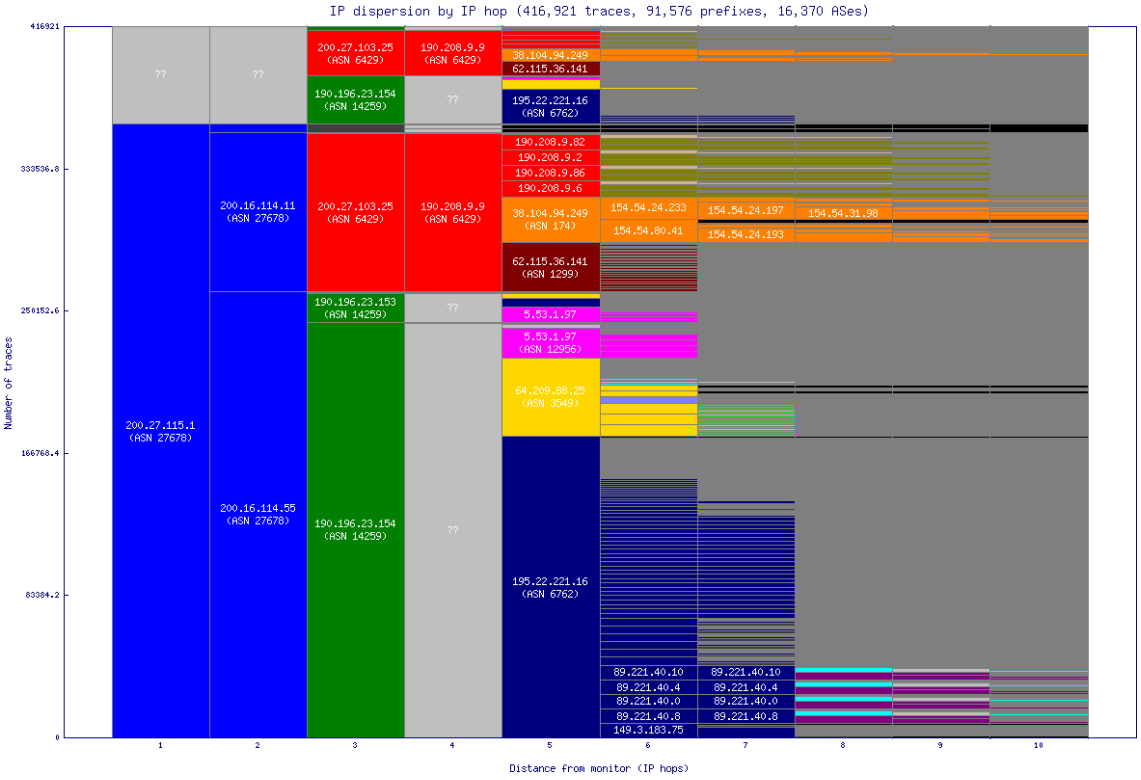


Figura 3.8 – Estadísticas medidor scl-cl, NIC Chile [58]

Siendo más específicos en los 4 primeros saltos:

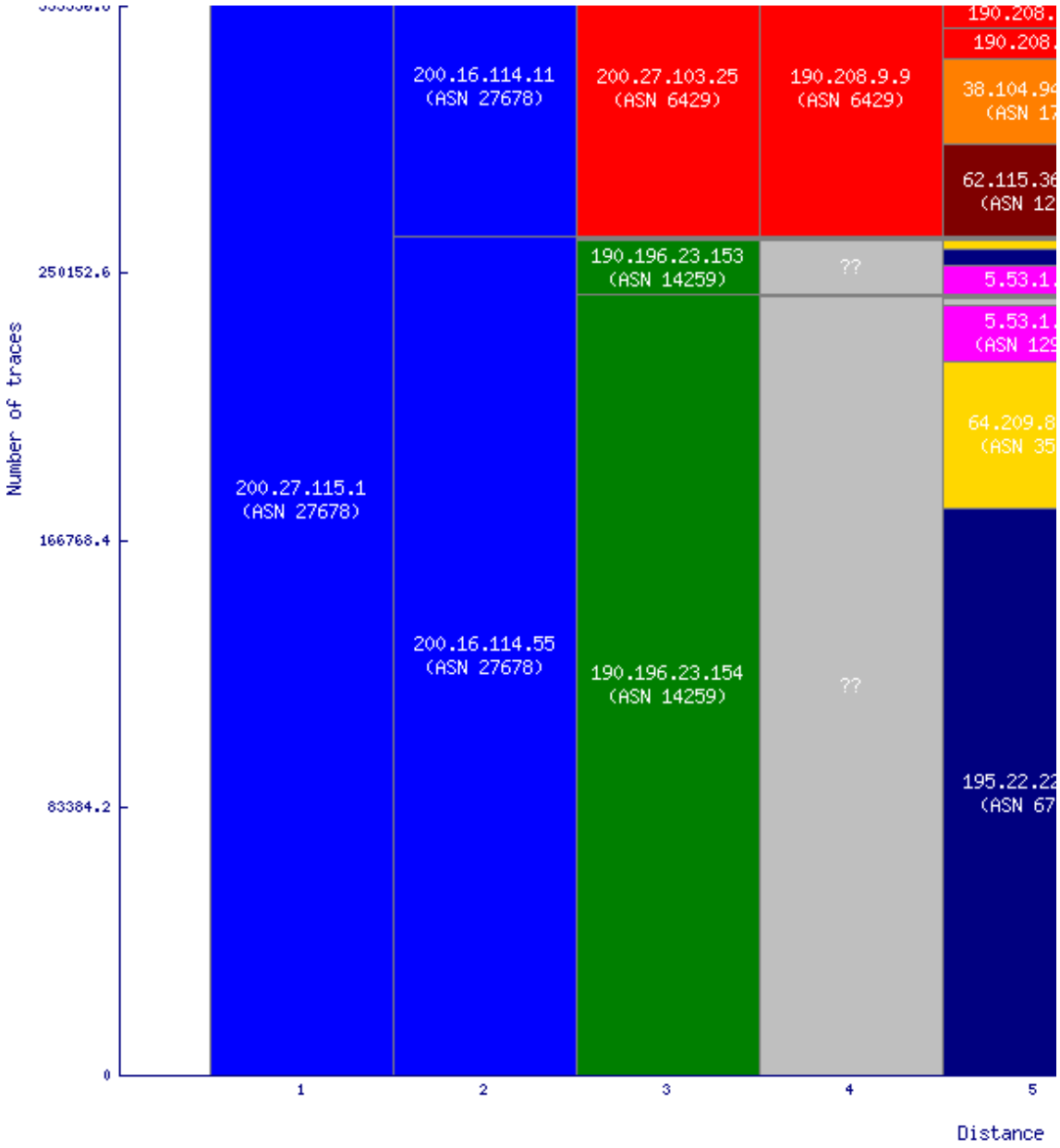


Figura 3.9 – Acercamiento Figura 3.8

Observando la cantidad medidas que son tenidas en cuenta, por encima de las 250.000, podemos decir que el patrón de comportamiento que existe en los 4 primeros saltos, desde el punto de vista de las latencias es significativo.
Por lo tanto a pesar de que las medidas entre los saltos intermedios no hayan sido directamente realizadas en la pruebas de CAIDA, el mecanismo indirecto que se tiene utiliza en esta etapa del proyecto para obtener medidas de latencia es válido.

Al igual que en el caso de RIPE NCC, la obtención de datos de CAIDA se realiza en la modalidad de etapas.

A diferencia de RIPE, CAIDA posee otro mecanismo de consulta de los datos almacenados. La misma es de acceso web, esquematizada con un formato del tipo árbol, ordenada en un primer nivel por el conjunto de medidores que conforman el equipo de prueba (*team-probing*), en un segundo nivel se ordena por años (de 2013 hacia atrás) y finalmente por ciclos de prueba. El formato es similar al utilizado por los servidores FTP (*File Transfer Protocol)*
A continuación se muestran las capturas de pantalla del mecanismo de consulta [59]:

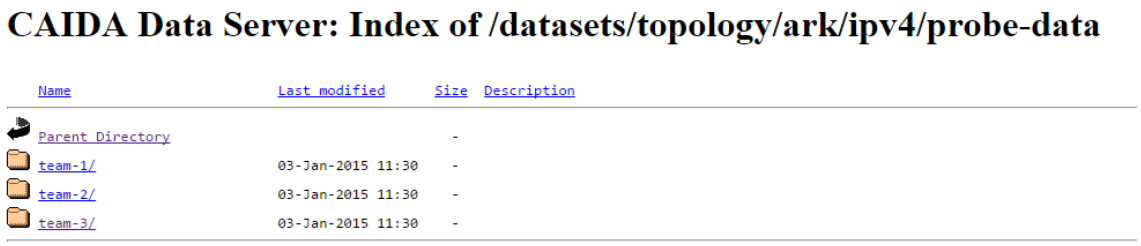


Figura 3.10 – Sección Teams

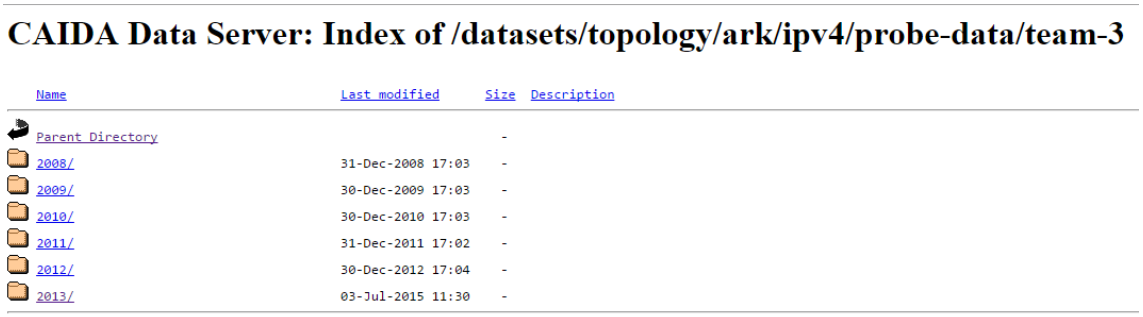


Figura 3.11 – Sección Años

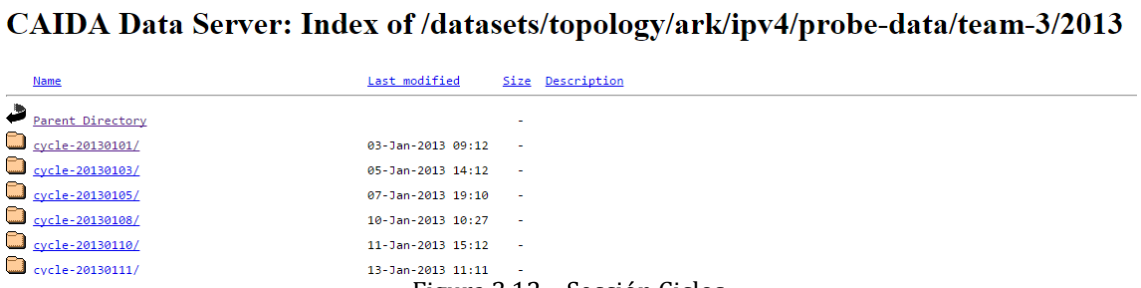


Figura 3.12 – Sección Ciclos

Dentro de cada ciclo, se encuentran las pruebas realizadas por cada medidor. Estas pruebas son generadas por una herramienta llamada *Scamper* [60].

Esta herramienta está diseñada para realizar pruebas a diferentes destinos de Internet, se realiza en paralelo para varios destinos al mismo momento, de manera de realizar pruebas en un tiempo razonable.
Al utilizar *Scamper* el resultado que se obtiene, es un archivo de tipo binario de extensión *wart.* Al igual que en otras aplicaciones de equipos de telecomunicaciones se utiliza el formato binario para ahorrar en memoria y en procesamiento. Para el caso de este proyecto y para realizar modelos similares entre CAIDA y RIPE, se utiliza una función dentro del paquete de *Scamper* llamada *sc_wart2json.* Esta función permite tomar el archivo binario y convertirlo en un archivo de tipo JSON, de manera de poder comprender el modelo de las pruebas de CAIDA y de poder manipularlo con scripts basados en *Python.*

Se puede resumir el proceso en el siguiente flujo:

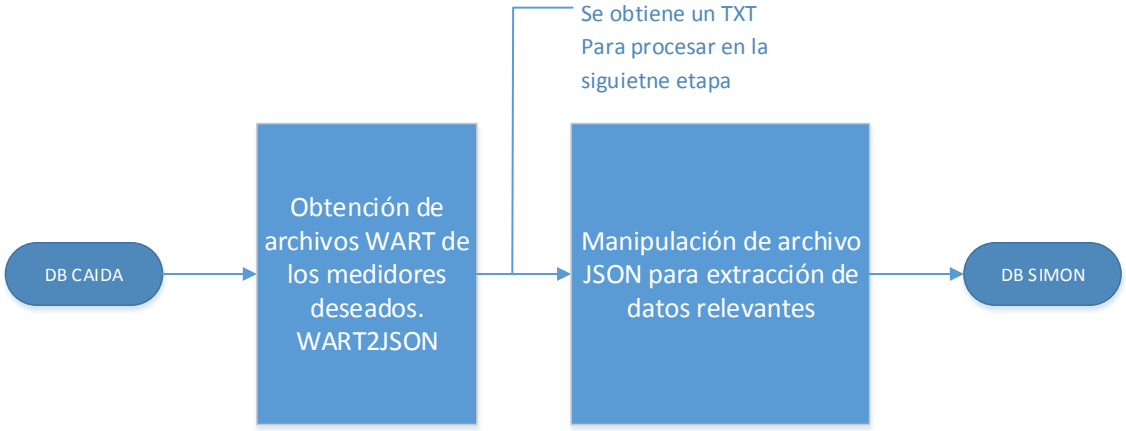


Figura 3.13 – Diagrama de etapas CAIDA

3.5.2 Etapas definidas para obtención de datos

ETAPA 1

En esta etapa se utiliza un primer script para obtener los datos correspondientes a los medidores ubicados en América Latina, en este caso los dos medidores de Chile y los dos de Brasil. Este script fue desarrollado por el equipo de programadores de LACNIC, por lo que no fue necesario cambiarlo en ningún sentido [61].
El script almacena los archivos del tipo .wart, que luego son consultados por el script de la siguiente etapa.

En este proyecto no nos centraremos en la programación de esta etapa, por lo que utilizaremos un archivo .*wart,* obtenido de uno de los medidores instalados en Chile, en el NIC Chile.

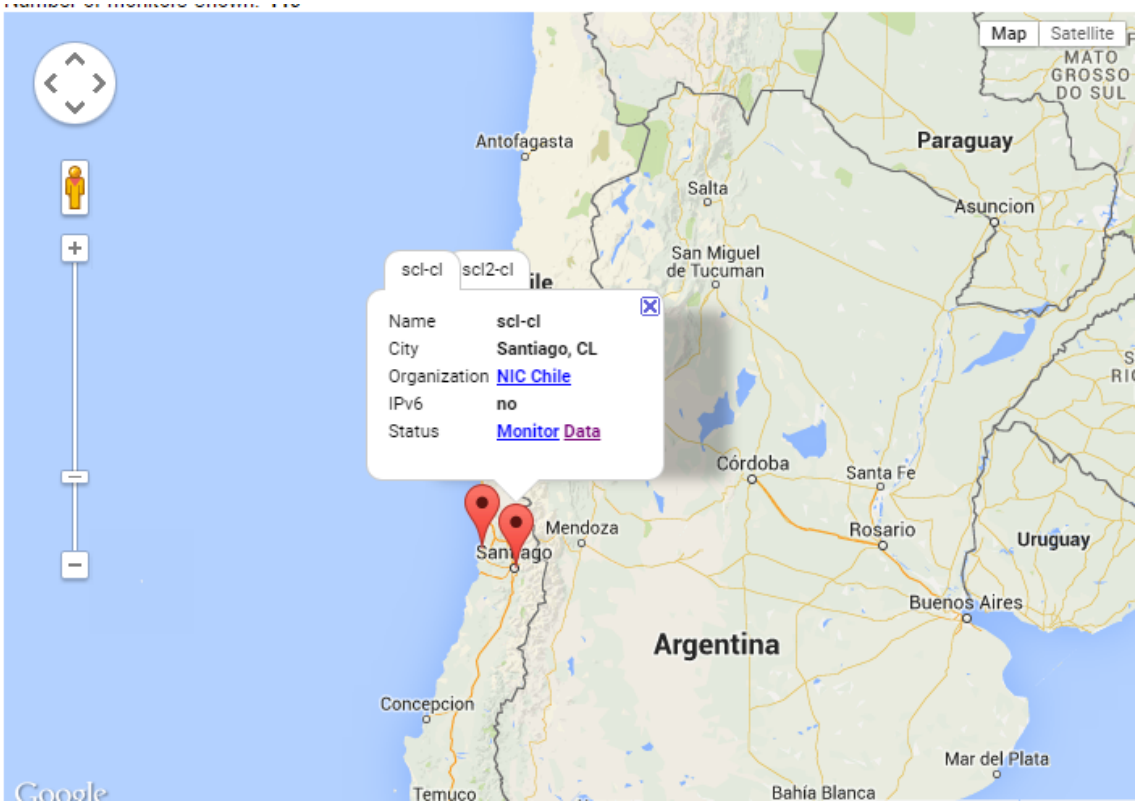


Figura 3.14 – Ubicación monitor scl-cl [54]

Este medidor aporta en un solo ciclo de aproximadamente 800Mb de información, por lo que se puede considerar suficiente para el alcance y pruebas de este proyecto.

ETAPA 2

En esta etapa se procede a tomar los archivos ya convertidos a JSON y manipularlos de manera de obtener los valores útiles para la base de datos de Simón.

A modo de ejemplo a continuación se puede ver el formato de una línea del archivo a ser utilizado:

{"version":"0.1","type":"trace", "userid":0, "method":"icmp-echo-paris", "src":"129.186.1.240", "dst":"71.192.213.124", "stop_reason":"COMPLETED", "stop_data":0, "start":{"sec":1354380947, "usec":350062, "ftime":"2012-12-01 11:55:47"}, "hop_count":12, "attempts":3, "hoplimit":0, "firsthop":1, "wait":5, "wait_*probe*":0, "tos":0, "*probe*_size":44, "hops":[{"addr":"129.186.6.251", "*probe*_ttl":1, "*probe*_id":1, "*probe*_size":44, "rtt":0.305, "reply_ttl":255, "reply_tos":192, "reply_size":56, "reply_ipid":6498, "icmp_type":11, "icmp_code":0, "icmp_q_ttl":1, "icmp_q_ipl":44, "icmp_q_tos":0},{"addr":"129.186.254.131", CONTINUA LINEA...}]}

En la sección del JSON correspondiente a los *hops* se puede apreciar las direcciones IP de los saltos por donde pasan los paquetes de prueba entre el origen y el destino.

Dado que la ETAPA 1 obtenemos los archivos correspondientes a los medidores en América Latina, queda asegurado que el origen es dentro del área de cobertura.

El siguiente ítem a tener en cuenta corresponde a *stop_reason.* En el mismo se detalla el estado de la prueba *traceroute.* Los estados que se pueden obtener de una prueba son los siguientes:

- *COMPLETED*: este estado establece que el *traceroute* se completó con éxito desde el origen al destino establecido
- *GAPLIMIT*: si se obtienen más de 5 saltos inalcanzables, el resultado queda almacenado con este estado
- *UNREACH*: este estado informa que el destino no puede ser alcanzado
- *LOOP*: el resultado de la prueba se guarda con este estado si un bucle es encontrado en el camino del *traceroute.*

Para esta sección se utilizan solamente los resultados que poseen el estado *COMPLETED,* a pesar de que los otros estados puedan aportar medidas útiles, de esta manera asegurando con el camino de origen a destino fue completado.

A continuación se destaca el campo *hops,* donde se detallan los saltos (camino IP) que realiza el *traceroute.* Cada uno de los saltos posee la información necesaria de latencia para ingresar a la base de datos de Simón.
Para el caso de CAIDA no se obtienen medidas de media o promedio en las latencias, por lo que se opta por almacenar la diferencia del contenido en el ítem *rtt* en los siguientes campos de la base de datos:

- min_rtt = dif
- ave_rtt = dif
- max_rtt = dif

Es posible que la diferencia entre *rtts* sea un valor menor a cero, por lo que estas medidas son descartadas.
Como ya fue mencionado, los resultados útiles para esta sección son aquellos contenidos en el territorio de acción de LACNIC, por lo que se realizan los chequeos correspondientes entre los saltos, para tomar solo aquellos que cumplen con asignaciones de los siguientes bloques /8:

bloques = ['177', '179', '181', '186', '187', '189', '190', '191', '200', '201']

Al igual que para el caso de RIPE, se realiza un chequeo de los países para cada una de las IP de los saltos útiles, por lo que se consulta la base de datos de geolocalización de RIPE para obtener esta información.

Finalmente se recurre nuevamente a la función *Results()* utilizada también para almacenar los datos de RIPE, y se almacena finalmente cada medida obtenida.
Dentro de la base de datos de Simón, podemos identificar las medidas obtenidas a través de este script, por el campo **testtype = 'caida_db' [62].**

Capítulo 4 - Incorporación de Sistemas Autónomos

4.1 Introducción

La base de datos de Simón está centrada en la obtención de medidas entre un origen IP y un destino IP. Esta desagregación es muy útil para un sinfín de propósitos, pero también se puede lograr un nivel de desagregación de similares características, a nivel de Números de Sistemas Autónomos (ASN –*Autonomus System Number*), lo cual permite ver información relacionada a las publicaciones BGP y al comportamiento de rutas en Internet

Los Números de Sistemas Autónomos, son grupos de redes IP los cuales se caracterizan por estar bajo la misma administración, la cual elije sus políticas de ruteo internas, se decide básicamente cómo se comportan los paquetes dentro de un mismo ASN. La parametrización por ASN es por lo tanto muy útil para identificar características de una determinada empresa o región [63].

En esta actividad se busca utilizar los datos ya obtenidos con los diferentes medidores, agrupándolos no por IP origen o destino, sino por ASN, de manera de obtener información alternativa para realizar análisis comparativos de latencias entre países o ASNs.

Durante el desarrollo de esta etapa se utiliza en una actividad inicial, la base completa de Simón y para una segunda actividad de utilizan las medidas aportadas por la base de datos de CAIDA.

4.2 Matriz de latencias por AS

El formato de la base de datos de Simón, se centra en información de orígenes y destinos IP y medidas de latencia entre las mismas. En esta primera actividad se busca tomar los datos IPs de esta base y utilizarlos para relacionar la latencia a los correspondientes ASNs.

Para construir la matriz e latencias se decide almacenar la misma en un archivo de texto, con formato JSON, de manera que pueda ser estudiada por otro script en caso de que se desee.
El formato del archivo será el siguiente:

{"Medida N": {"asn_origin ": "7303", "as_list": {"ASN 27896": [203], "ASN 6147": [431, 544, 200],..., [OTROS DESTINO] }}, "Medida N+1":{...}}

Donde se identifica inicialmente una medida y un ASN de origen. Luego se completa una lista con las respectivas latencias a los diferentes ASN de destino. En caso que un destino se repita, la lista correspondiente a ese destino, contiene todas las latencias.

Para implementar estos requerimientos se deben seguir los siguientes lineamientos en la programación de script correspondiente a esta etapa:

- Consultar la base de datos
- Definir ASN origen y lista de ASNs destino
- Escribir el archivo de texto en formato JSON son las características mencionadas

Como ya fue mencionado en varias oportunidades en el texto la base de datos de Simón es extensa, por lo que realizar un script que realice múltiples consultas a la base de datos, o consultas de baja performance, resulta poco beneficioso en cuanto a tiempo y capacidad de procesamiento. Por lo tanto, el formato y las características de la consulta SQL realizada, son de vital importancia en este caso.

Hay que destacar que solo se necesitan ciertos campos de la tabla *simon_app_results*, contenedora de la información, que son utilizados en esta sección:

- *Ip_origin*
- *Ip_destination*
- *Ave_rtt*

A partir de ellas se genera la consulta SQL de manera de obtener, en primer lugar, las IP de origen de las pruebas en manera ordenada (creciente o decreciente). Luego es importante que la consulta ordene las IP de destino, obteniendo así una lista que

debe ser recorrida una única vez para obtener los datos de todas las medidas realizadas.
Dicho en otras palabras, no se debe recorrer toda la base buscando orígenes repetidos, ya que al encontrar un origen, todos los que le siguen (en caso de que los haya) son del mismo origen. A su vez ordenando por destino, los que se encuentran repetidos son nuevamente seguidos, permitiendo así completar las listas de destinos más fácilmente.

Se puede resumir el comportamiento del script [64] con el siguiente pseudocódigo:

CONSULTO TABLA simon_app_results ordenadamente
CONSULTO primer origen
WHILE hay_medidas
    IF se repite origen
        Guardo en lista
        IF se repite destino
            Guardo en lista correspondiente
    Almacena en archivo TXT

4.2.1 Comentarios

Durante la programación y testeo del script surgieron algunas situaciones que son importantes de mencionar.

En primer lugar, al realizar el chequeo del archivo de texto con los resultados correspondientes, se observó que el ASN 15576, se repetía en varios de los resultados. Buscando dentro de la base de datos de ASNs de Simón, ese ASN corresponde a un ASN por *default.* Esto significa que al momento de almacenar una nueva medida en la base de datos, si no se conoce cuál es el ASN que publica el bloque, que contiene finalmente la IP de la prueba, se le adjudica el ASN 15776.
Al corroborar que varios de los resultados con el ASN *default* están siendo anunciados vía BGP por otros ASN, se tomaron dos iniciativas:

1. Construir una tabla alternativa en la base de datos de Simón con los ASN que efectivamente anuncian las IPs en los registros. [65]
2. Realizar una actualización general de la tabla de ASNs provista por Simón. Esta tabla se actualiza en función de los datos del *RIS (Routing Information Service)* de RIPE [66].

A pesar de que el script de la opción 1 fue realizado y funciona correctamente, se optó realizar una actualización en la base de datos de producción, de manera de impactar también en el resto de las aplicaciones de Simón.

Por otro lado el script realizado se utiliza como herramienta de chequeo, para maquetas de prueba o en caso de querer obtener muestras del estado del *RIS* de RIPE.

4.3 Relación largo de AS-Path y latencia

Teniendo en cuenta la cantidad de opciones que poseen los datos para ir de un origen a un destino, desde el punto de vista de la performance, es importante saber si un camino es mejor que otro.

En esta etapa del proyecto se busca identificar la existencia de una relación entre la latencia y el camino que sigue la información desde el punto de vista del protocolo *BGP* (as-path).
Para este fin se utiliza la información recabada de los medidores de CAIDA. Para este caso y dadas las características de medición de CAIDA, ya mencionadas, se utiliza información de tres ciclos distintos de uno de los medidores instalados en Chile. Esto permite obtener ASN destinos repetidos.

Al igual que en secciones anteriores se busca crear archivos con formato JSON y se trabaja en etapas, de manera de filtrar en cada paso la información no relevante.
En este caso estaremos trabajando con 3 archivos de texto, que inicialmente poseen un tamaño de 811Mb, 404Mb y 730Mb respectivamente, lo que hace que dentro de los ambientes de pruebas configurados, sean archivos difíciles de manejar. En la Figura 4.1 se detalla el procedimiento de manera general:

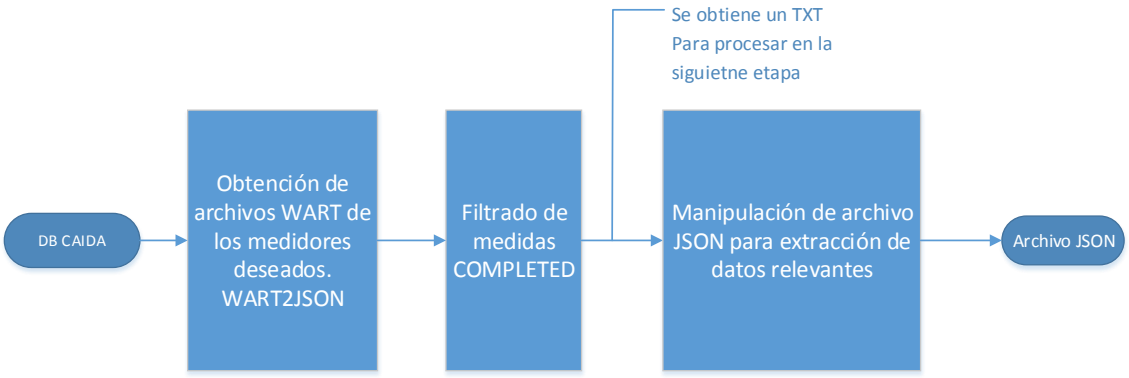


Figura 4.1 – Diagrama de flujo archivo AS-path

En una primera etapa, se obtienen los archivos *wart* de los repositorios de CAIDA, seguidamente se procede a convertirlos a formato JSON para ser manipulados.

Luego en una segunda etapa se desarrolla un script [67] que lee estos archivos uno a uno, almacenando en un archivo de salida, solo los resultados de las pruebas que posean el status *COMPLETED.*

Un seudocódigo básico para esta etapa es el siguiente:

```
FOR cantidad de archivos en carpeta
    WHILE existe línea
        IF stop_reason == "COMPLETED"
            Guardo en archivo de salida
```

En una tercera etapa se procede a interpretar el archivo de salida obtenido en la etapa anterior, leyendo línea a línea las pruebas, procesando solamente aquellas que tengan como destino final una IP comprendida en la región de LACNIC.
Asegurando que la IP sea la indicada, se procede a buscar el ASN correspondiente a esa IP. Luego se busca en todo el archivo por otras IPs con el mismo ASN de destino. De esta manera se generan listas de caminos y las respectivas latencias en cada uno de los *hops* del camino. Se realiza a su vez un chequeo, para evitar repetir resultados ya guardados.

Al igual que en otras etapas del proyecto, se desarrollaron dos tipos distintos de consultas de ASNs según la IP. En un primer caso se apunta a buscar los ASNs desde la base de datos de RIPE mediante una consulta web [68], pero esto hace que la performance del script disminuya considerablemente.
En una segunda instancia se utiliza la base de datos de ASNs de Simón, que a pesar de no estar actualizada al momento de realizar las pruebas, posee mucho más performance, permitiendo obtener resultados más rápidamente.

Se puede resumir el comportamiento del script [69] en el pseudocódigo a continuación:

```
CARGO archivo etapa 2
LEO línea del archivo
WHILE hay línea:
    IF pertenece a LACNIC y ASN no chequeado aún:
        GUARDO puntero con posición en archivo
        WHILE hay línea:
            IF ASNs iguales
                Guardo en Lista
            LEO línea
    GUARDO listas creadas en archivo final
    OBTENGO puntero
    LEO línea a partir de puntero
```

Una muestra del archivo de salida final obtenido es la siguiente:

{"IPs2": ["200.27.103.25", "190.208.5.14", "190.208.9.13", "157.238.179.17", "129.250.2.128", "204.255.168.13", "152.63.0.234", "152.63.97.98", "157.130.155.114", "201.125.58.77", "189.189.102.152"], "IPs3": ["200.27.115.1", "200.27.103.25", "190.208.5.14", "190.208.9.13", "157.238.179.17", "129.250.2.128", "204.255.168.13", "152.63.112.13", "152.63.118.97", "208.222.15.166", "201.125.56.62", "189.246.1.30", "189.246.1.205", "201.125.70.126", "201.125.70.121", "201.154.136.5", "189.189.126.51"], "IPs1": ["200.27.103.25", "200.27.5.106", "190.208.9.13", "157.238.179.17", "129.250.2.128", "204.255.168.13", "152.63.96.250", "152.63.97.86", "157.130.155.94", "201.154.136.249", "189.189.121.215"], "IPs4": ["200.27.103.25", "200.27.5.106", "190.208.9.13", "157.238.179.17", "129.250.2.128", "204.255.168.13", "152.63.112.13", "152.63.118.57", "63.114.61.82", "201.125.56.110", "189.246.1.26", "189.246.1.174", "189.246.1.14", "201.125.72.73", "201.154.136.249", "189.189.113.71"], "RTTs2": [0.618, 0.951, 1.331, 107.743, 107.919, 136.942, 172.759, 172.815, 236.352, 264.499, 280.058], "RTTs3": [0.146, 0.539, 0.911, 1.164, 107.722, 107.649, 136.872, 230.039, 234.812, 211.216, 226.212, 228.185, 226.678, 227.402, 225.375, 223.796, 236.628], "RTTs1": [0.524, 0.884, 1.248, 107.771, 108.187, 137.08, 172.312, 172.32, 236.85, 264.307, 278.288], "base_asn": "8151", "RTTs4": [0.545, 0.857, 1.266, 107.68, 107.814, 136.824, 204.02, 204.166, 206.387, 227.582, 227.216, 227.394, 226.085, 226.216, 223.286, 232.599]}

Se puede observar resaltados los caminos IP de la medida, sus respectivas latencias y el campo *base_asn,* que posee como valor el ASN de destino para todos los caminos dentro de la línea.

Este archivo puede ser luego interpretado con otro script [70], para realizar las visualizaciones que se crean pertinentes. En este caso se optó por utilizar un script que muestre resultados en un *shell* tipo *Unix,* para una visualización en formato de texto. Esto es el paso inicial para ser llevado luego a un formato visual más amigable.

Capítulo 5 - Incorporación de Puntos de Intercambio de Tráfico

5.1 Introducción

El crecimiento de Internet, la necesidad cada vez mayor de conectividad de mejor calidad y mejores precios, han sido los focos a los que apuntan, organizaciones vinculadas a Internet, en la instalación y promoción de los puntos de intercambio de tráfico en todo el mundo.

Los Puntos de Intercambio de tráfico o *IXP (Internet eXchange Points)* son infraestructura en la red de los países, que permite el intercambio de información entre empresas u organizaciones dentro de un mismo país, evitando enviar información a través de redes exteriores al país, para luego volver a ingresar al mismo.
Los *IXP* son por lo tanto muy importantes a la hora de mejorar la calidad y la estabilidad de las redes en un país. Es por esta razón, que a través de este aporte, se busca medir la calidad de las redes que intercambian información en un IXP, a aquellas que no lo hacen [71].

En esta etapa, se utiliza la base de datos de Simón y la información contenida en la base de datos de PeeringDB [72], un repositorio donde se puede encontrar la información sobre los *IXPs,* en lo que respecta a redes y *peers* conectados a determinada infraestructura.

5.2 Comparación de medidas

En esta actividad, se apunta a desarrollar aplicaciones y mediciones, para entornos de pre-producción que involucren infraestructura de *IXPs*, por lo que se tienen ciertas consideraciones en la realización de las herramientas utilizadas y desarrolladas. En primer lugar se apunta a analizar medidas de un solo país, como etapa de desarrollo inicial. El país elegido es Ecuador, ya que reúne las siguientes características:

- Posee un *IXP*
- Varias organizaciones intercambian información a través del *IXP* (son *peers)*
- Algunas organizaciones NO pertenecen al *IXP,* lo que permite realizar las comparaciones entre organizaciones pertenecientes o no.

Por otro lado, dado que no se tiene información de cual es efectivamente el tráfico intercambiado por cada organización (esto se negocia en cada *IXP* de manera independiente), se considera que la mayoría del tráfico intercambiado es efectivamente intercambiado en el *IXP,* por lo que si las medidas de la base de datos de Simón pertenecen a un ASN dentro del IXP, se considera que la información hacia otro ASN del IXP es intercambiada directamente. A priori esta asunción puede no ser general para todo el tráfico intercambiado, pero es una buena aproximación para la construcción inicial de las herramientas de medición.

Respecto a la integración con el ambiente de producción de LACNIC, para realizar medidas en todos los *IXPs* de la región, se deben realizar modificaciones a la tabla principal de resultados de Simón. Esto implica trabajo a realizar por los desarrolladores de LACNIC, que no será tenido en cuenta en este proyecto. El cambio principal a realizar por LACNIC, es agregar un campo a la tabla principal de resultados, con un identificador de *IXP.* Si el origen o el destino (o ambos) pertenecen a un *IXP*, el nuevo campo contendrá el id especifico de ese punto de intercambio de tráfico, que hace referencia a la tabla general de *IXPs.*

Teniendo estas consideraciones en cuenta se desarrolla un script [73] preliminar con funcionalidades que apuntan a estudio, por el momento, de un solo país.
En primer lugar se obtiene la información de los puntos de intercambio de tráfico en Ecuador, a través de la plataforma *PeeringDB,* mediante su página web. Esta plataforma acepta el ingreso como invitado para obtener información de todos los *IXPs* registrados.

En la figura 5.1 se muestra una captura de pantalla con la interfaz gráfica que utiliza esta herramienta, filtrado específicamente por país y por región:

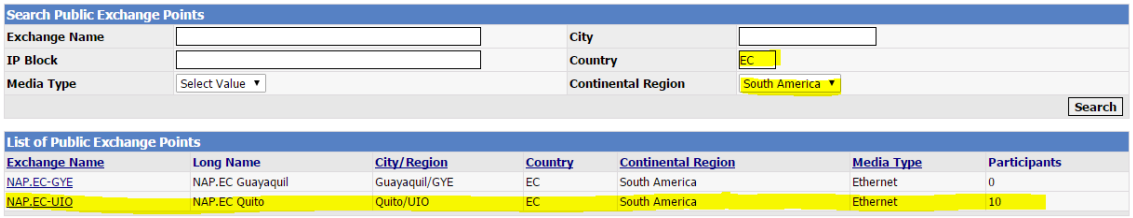


Figura 5.1 – Búsqueda IXP filtrado por país Ecuador

El enfoque se realiza en el *IXP* NAP.EC, donde se pueden encontrar los siguientes *peers:*

List of Peers at this Exchange Point (Total: 6)

Peer Name	Local ASN	IP Address	IPs	Policy
AEPROVI	27814	200.1.6.29	1	Open
Internexa	27765	200.1.6.21	1	Open
ISC	27916	200.1.6.15	4	Open
l.root-servers.net	20144	200.1.6.23	2	Selective
Telconet	19169		1	Selective
WISP Ecuador	52482	200.1.6.24	1	Open

Figura 5.2 – Participantes del IXP elegido

Mediante esta interfaz, no se pueden obtener las características de los *peers* de manera automática con una plataforma REST, por lo que los datos deben ser recolectados a mano al momento de realización de esta etapa.
Es importante destacar que *PeerinDB* está desarrollando una plataforma que permitirá la obtención de datos con métodos REST, pero aún se encuentra en una versión *beta* y la información contenida es solamente de muestra [74].
Teniendo esta situación en cuenta, durante la etapa de desarrollo del script, también se procede a crear las tablas en la base de datos de Simón, que almacenarán los datos de los *IXPs*.
De esta manera se avanza en la solución final para integrar todos los países en el ambiente de Simón, en referencia a puntos de intercambio de tráfico.

En lo que respecta al script desarrollado, la metodología utilizada es de "función", lo que permite ingresar un parámetro al momento de ejecutar el script. De esta manera se establece como parámetro, la nomenclatura ISO de los países, por ejemplo 'EC'.
A su vez, se desarrollan algunas funcionalidades para el futuro, como puede ser la consulta de a que *IXP* pertenece una determinada IP (en el caso que así sea) (get_as_ixp()). Esta funcionalidad se basa en los desarrollos futuros de LACNIC.

A continuación se detalla un seudocódigo, con las características principales del script desarrollado:

ELIJO país
CONSULTO db de Simón por país
FOR cantidad de resultados de la base:
    IF origen_ixp==destino_ixp:
        GUARDO en lista de resultados dentro de IXP
    ELSE:
        GUARDO en lista de resultados fuera de IXP

GENERO archivo JSON con resultados

Al igual que en otras instancias, se almacenan los resultados en un archivo tipo JSON, que luego puede ser interpretado por otro script. Para el caso de este proyecto, se desarrolló un simple script de análisis [75] que compara las mediciones internas al *IXP,* con las externas al mismo.

Capítulo 6 - Modificaciones en Plataformas actuales

6.1 Introducción

Para mejorar el abanico de medidas, se necesita identificar cuáles son algunas de las necesidades dentro de las medidas ya existentes. Al identificar estas necesidades se puede enfocar mejor los esfuerzos en la obtención de medidas de latencia.
En esta etapa, se busca identificar algunas de esas necesidades e implementar herramientas que permitan realizar una mejor gestión de las medidas a recabar.

Como fue mencionado inicialmente en este documento, el método más utilizado para obtener medidas en el proyecto es a través del medidor *JavaScript,* por lo que se procederá a realizar un script, basado en el comportamiento de la función de obtención de puntos de testeo ya desarrollado por LACNIC, con algunos cambios en el modo que se manejan las pruebas.

6.2 Modificaciones realizadas

Mediante el script desarrollado [76] se busca tener alternativas al comportamiento estándar del medidor *JavaScript.* Para esto se modifica una de las funciones claves del mismo que obtiene los puntos de testeo a los cuales se les realiza las pruebas de latencia. Por defecto la función devuelve N cantidad de puntos de testeo en forma aleatoria, compuesto de servidores de *SpeedTest* distribuidos en la región de LACNIC.

Antes de realizar las modificaciones, es necesario establecer cuáles son los criterios a tener en cuenta, de manera de obtener medidas de mejor calidad, a través de la elección inteligente de los puntos de testeo. Para esto se tiene en cuenta los siguientes criterios:

- País de origen y destino de la prueba
- Cantidad de pruebas realizadas desde o hasta ese país
- Mantener o no los puntos elegido aleatoriamente

A partir de los criterios mencionados se establecen las siguientes líneas de acción para elegir los puntos de testeo:

- En función al país de origen de la prueba, se elige al menos un servidor de SpeedTest contenido en ese país. En caso de no haber ninguno, se elige uno aleatoriamente
- Teniendo en cuenta el país de origen, se calcula la cantidad de medidas a cada país y se toma como destino de prueba, el país con menores medidas.
- El resto de los puntos necesarios se eligen de manera aleatoria.

Estos lineamientos permiten obtener medidas con finalidades muy definidas, independizando así el código de los algoritmos de aleatorización utilizados por el lenguaje de programación utilizado.

A pesar de que los criterios previamente descritos son los que se utilizarán en esta etapa, se pueden realizar modificaciones muy simples a cada uno de ellos, variando así el resultado de las pruebas realizadas. La importancia de poseer versatilidad al momento de hacer las pruebas, permite adaptaciones a situaciones o momentos muy puntuales.

A continuación se detalla un pseudocódigo de la solución implementada, teniendo en cuentas los criterios analizados.

INICIO (ip, país de origen, cantidad de servidores)
OBTENGO servidor de test del país de origen de la prueba (consulta SQL)
GENERO lista con la cantidad de pruebas del país de origen a cada país de LACNIC
BUSCO el mínimo de la lista obtenida previamente
BUSCO destinos aleatorios (cantidad de servidores – min – servidor país)
CREO lista con todos los destinos buscados
FOR ítem en la lista
    Obtengo información de punto de testing
    GUARDO información en formato JSON

DEVUELVO lista
END

La lista que devuelve esta función, proceso es consultada mediante un REST, de la misma manera que lo hace las aplicaciones RIPE. Para el caso de la maqueta utilizada la url a consultar debe tener el siguiente formato:

http://localhost:8001/web_points/4/

Donde el número 4, es la cantidad de servidores de prueba solicitados.

Capítulo 7 - Índice de conectividad

En esta actividad se pretende estudiar la posibilidad de establecer un índice, en principio numérico, que defina la calidad de los enlaces de Internet de un determinado país, un país con respecto a otro, e inclusive entre ASNs.

Dada la complejidad y sensibilidad que puede generar la creación y publicación de un índice, se pretende solamente definir algunos lineamientos, teniendo en cuenta la nueva información con la que se cuenta, gracias a los aportes descritos previamente en este documento. Es importante destacar que LACNIC, como organización administradora de las direcciones IP, tiene el deber de establecer procesos de análisis claros y de índole publica, para evitar debates, que afecten el correcto relacionamiento entre países u organizaciones de su región de influencia.

El índice debe ser construido en base a resultados o medidas al alcance del proyecto Simón, de manera de no depender de ningún agente externo para evitar restringir el marco de estudio del índice, y poder revisar el mismo de manera periódica. En este sentido se eligen los siguientes lineamientos para diagramar el índice:

- *AS-hop count:* este factor define la cantidad de ASNs por los cuales pasa la información y en combinación con la lejanía o cercanía geográfica resulta bastante representativo de un aspecto de la calidad de los enlaces. Dentro de este ítem se considera también, el pasaje de la información por rutas anunciadas por ASNs fuera de la región LACNIC, lo cual puede influir tanto positiva como negativamente.
  A su vez, la variación de la cantidad de saltos en un intervalo de tiempo, muestra la estabilidad dada para un ASN particular, que en ciertos casos podrá tomarse como valor de referencia para un determinado país. En este sentido es importante tener en cuenta que la variación mencionada afecta en mayor medida a países que poseen pocos proveedores de conectividad, o al menos un proveedor muy dominante en el mercado.
- Situación geográfica: dada la diversidad de entornos en la región, se detecta como un factor importante la situación geográfica de un país. Por ejemplo, dependiendo si un país es interno al continente (Paraguay, Bolivia, etc.), tiene costa o es una isla, eso favorece o dificulta la conectividad internacional. También la cercanía a puntos que concentran gran cantidad de tráfico como ser Estados Unidos o Europa es un factor importante a tener en cuenta.
- Relevamiento de las medidas de latencia: en función de los resultados de latencia obtenidos y las herramientas dentro del proyecto, establecer calidad de los enlaces, entre países o entre ASNs
- Presencia de *IXPs*: la presencia de puntos de intercambio de tráfico, más aún en países con grandes extensiones, como Argentina o Brasil, puede ser un factor importante a la hora de definir la conectividad(o calidad de conexión que perciben los usuarios finales) que posee un país o región.
- Penetración de Internet: es importante observar no solamente la calidad de los enlaces de un país o región, sino también el alcance de la conectividad en lo referente a nivel de cobertura de usuarios. Por ejemplo, si una región posee

excelente calidad de conexión en las ciudades principales, o solo para un determinado estrato de la población, esto influye negativamente en la penetración Internet. Existen varias organizaciones que dedican parte o totalmente sus esfuerzos a generar estadísticas, índices e indicadores de penetración, velocidades de acceso, precio medio del acceso a Internet, etc., accesibles algunas en forma paga y otras en forma gratuita.

Las relaciones y el peso de cada uno de los factores deben ser aún estudiado con mayor detalle. También debe considerarse que la lista descrita, se encuentra abierta a modificaciones o adición de ítems. Por ello se plantea y propone el avanzar en las discusiones sobre este tema en sucesivos aportes al proyecto Simón, ya que un índice, es una herramienta de gran utilidad, tanto para el sector empresarial, como para gobiernos y sociedades, al momento de definir la calidad de las redes en un país o región.

Estos factores y su influencia serán nuevamente tenidos en cuenta en las conclusiones.

Capítulo 8 - Resultados

Como se mencionó desde un comienzo, este proyecto tiene como objetivo principal ampliar y mejorar las medidas y las herramientas contenidas dentro del proyecto Simón. Con este fin se realizó una planificación con varias etapas que impactan en distintas secciones del proyecto. A continuación se detallan los resultados, en relación a los objetivos planteados.

Actividad 1 - Familiarización con las herramientas de trabajo y el entorno del proyecto

La actividad referente a las herramientas y entorno del proyecto, tuvo dos grandes etapas. La primera consistió en aprender y poder desarrollar en nuevos lenguajes de programación como, *Python, JavaScript, SQL* y ambientes de trabajo generalmente asociados al desarrollo ágil de páginas web, como lo es *Django.*
La segunda etapa consistió en ahondar en la programación específica de los ambientes del proyecto Simón, comprendiendo la arquitectura y la lógica de las herramientas ya desarrolladas, para poder luego modificarlas de manera correcta.
En ambas etapas se obtuvieron los resultados esperados, adquiriendo los conocimientos necesarios para afrontar el resto de las actividades proyectadas en la contribución.

Actividad 2 - Normalización de datos

El segundo de los objetivos planteados en el proyecto consistió en la normalización de los datos. En este caso se realizó una división en dos actividades distintas, la primera buscando normalizar los datos obtenidos por los distintos medidores y la segunda buscando detectar y normalizar los efectos en las medidas de latencia, relacionados a las diferentes sistemas operativos y navegadores.
En el transcurso del proyecto se optó por comenzar por la segunda actividad, ya que se sabía insumiría más tiempo.
Como resultado de esta etapa se consiguió agrupar y desarrollar un conjunto de herramientas que permite la normalización de las medidas, en función del sistema operativo y navegador con los que fueron realizadas. A pesar de que algunas pruebas no pudieron llevarse a cabo, por incompatibilidades entre algunos elementos de software (en color amarillo), los resultados obtenidos son relevantes
A partir de la Tabla 2.6 de resultados, se definen los siguientes porcentajes (Tabla 8.1) de corrimiento con respecto a la base elegida:

	Porcentaje de corrección			
	Windows 7	**Windows 8.1**	**Mac OS**	**Windows XP**
Chrome	BASE	9%	39%	17%
Firefox	22%	17%	17%	374%
Safari	13%	4%	65%	0%
IE	13%	13%	0%	0%
Opera	22%	4%	0%	30%

Tabla 8.1 – Porcentajes de corrección

A partir de estos porcentajes se puede realizar un correcto análisis del impacto de los sistemas operativos y los navegadores en las medidas de latencia.

En la sucesiva actividad contenida en el mismo capítulo, se trabajó con los cuatro medidores *TCP, NTP, Javascript e ICMP.*
Todos ellos poseen el mismo fin, medir latencias, pero lo hacen con mecanismos diferentes, por lo que se comparó cada uno de ellos, con la intención de buscar, al igual que en el caso de los navegadores y sistemas operativos, una capa de desarrollo que normalice los resultados de cada uno de manera de obtener resultados independientes de la herramienta de medida que se utilice.
Luego de analizar los mecanismos de trabajo, ventajas, desventajas y utilización dentro del proyecto, se puede decidir si generar o no, la capa de desarrollo que normalice los resultados.
El análisis fue correctamente realizado, tomándose la decisión de no generar la capa de normalización. A continuación en el capítulo de conclusiones se detalla a fondo esta decisión.

Actividad 3 - Incorporación de mediciones de *partners* internacionales

En esta actividad se trabajó en la incorporación de las mediciones, obtenidas de proyectos de las organizaciones internacionales RIPE NCC y CAIDA. En el transcurso del proyecto, esta actividad tuvo como resultado uno de los entregables más complejos y demandantes, por lo que se decidió comenzar con esta actividad en primer lugar, a pesar de no ser la primera en el cronograma proyectado al comienzo de la contribución.
Durante el desarrollo de la solución para RIPE, se planteó inicialmente una línea de trabajo, que involucraba realizar la extracción de los datos con un solo script, camino que fue luego descartado, ya que las pruebas de maqueta insumían mucho tiempo. Esto se debe, como fue mencionado en el desarrollo, a las consultas web que deben realizarse para obtener los datos. Dada esta situación se decide trabajar en etapas, la primera de las cuales, extrae una gran cantidad de datos de la base de datos de RIPE con muy poco filtrado, para luego ser manipulada localmente en los ambientes de maqueta. Dado que este resultado fue muy beneficioso, se decide usarla también para el caso de CAIDA. El mismo se constituyó como un buen modelo de trabajo en

etapas, aplicable y replicable en caso futuro de querer obtener información de otra organización por fuera de las ya trabajadas.
Por el lado de RIPE, luego de realizar los trabajos de filtrado, se obtuvieron algo más de 1700 medidas de latencia. En cambio, en lo que respecta a CAIDA, se obtuvieron por encima de las 20.000 medidas, teniendo en cuenta un solo medidor en un periodo de tiempo de 48 horas. Por lo tanto, es posible disponer aún de una mayor cantidad de medidas de CAIDA, teniendo en cuenta más ciclos de pruebas, de los considerados en este proyecto.

Actividad 4 - Incorporación de sistemas autónomos

En lo que respecta a la incorporación de medidas entre sistemas autónomos, el proyecto relacionó la latencia con los conjuntos de redes bajo un mismo ASN, en lugar de hacerlo entre destinos IP.
Para lograr este fin se proyectaron 3 actividades: generar una matriz de latencias en base a ASNs, estudiar la relación entre *AS-path* y la latencia y realizar un diagrama de grafos.
Luego de comenzar con la actividad, se desestimó la realización de un diagrama con componentes visuales (estudio de grafos) y se optó por realizar listas en formato JSON que contengan la información requerida. Se prevé realizar una visualización amigable de las listas en el futuro, posiblemente dentro de la página web de Simón y dado que las herramientas para crear las listas ya fueron desarrolladas, el pasaje a formato visual web es más sencillo.
En cuanto a la matriz de latencias y ASNs, como fue mencionado en el desarrollo de la actividad, se plantearon dos opciones, una basada en herramientas dependientes de RIPE y otra, menos actualizada, dependiente de las herramientas de Simón. Finalmente se optó por tomar las herramientas de Simón, que poseen una mejor performance y realizar una actualización de la base de datos del proyecto, ya que la gran cantidad de datos procesados, superó las capacidades proyectadas de la maqueta, siendo posible procesar de manera parcial los datos contenidos en la base. Finalmente se desarrolló como resultado el script correspondiente para la correcta generación de la matriz de latencias en función a Números de Sistemas Autónomos (ASNs).

A continuación se muestra de manera parcial la matriz generada:

{"Medida 2": {"asn_origin ": "22047", "as_list": {"ASN 16960": [281], "ASN 22047": [1995, 1996], "ASN 4230": [0, 0, 0], "ASN 11556": [1980, 1982], "ASN 6471": [0], "ASN 28284": [1999], "ASN 27768": [1980], "ASN 19182": [1981, 1978], "ASN 28114": [1985], "ASN 11581": [1977], "ASN 28554": [2000], "ASN 6332": [1996], "ASN 8151": [1990, 1997, 1985, 1984, 1980], "ASN 27724": [1980], "ASN 28652": [1993], "ASN 28555": [2000], "ASN 11844": [0, 0], "ASN 28539": [0], "ASN 28094": [1989], "ASN 22566": [1994], "ASN 1916": [2001, 1987, 1993], "ASN 28512": [2000, 1998], "ASN 22250": [1989, 1993], "ASN 17072": [2000], "ASN 11830": [1983, 1975], "ASN 263114": [1992], "ASN 28589": [1987], "ASN 8167": [1991, 1995, 1975], "ASN 6568": [0, 0], "ASN 6400": [1980], "ASN 27927": [1979, 1999],

"ASN 18809": [1981], "ASN 22908": [1978], "ASN 11172": [1972], "ASN 28549": [1997, 1981], "ASN 28270": [1985], "ASN 32098": [1981, 1982], "ASN 14868": [466, 1987], "ASN 28606": [1975, 1981], "ASN 16735": [1982], "ASN 13999": [2001], "ASN 28509": [1994], "ASN 28524": [1996]}}, "Medida 0": {"asn_origin ": "7303", "as_list": {"ASN 27896": [203], "ASN 6147": [431], "ASN 13489": [203], "ASN 22047": [150], "ASN 14187": [181], "ASN 4230": [259], "ASN 11284": [101], "ASN 11556": [174], "ASN 27956": [197], "ASN 27742": [212], "ASN 19180": [146], "ASN 27768": [53], "ASN 2716": [165], "ASN 26610": [166], "ASN 6332": [237], "ASN 8151": [303], "ASN 16814": [15], "ASN 263227": [396], "ASN 27727": [375], "ASN 3549": [190], "ASN 27761": [219], "ASN 18747": [237], "ASN 16531": [231], "ASN 14754": [215], "ASN 22884": [228], "ASN 22548": [187, 268], "ASN 65003": [68], "ASN 20299": [184], "ASN 27884": [246], "ASN 11664": [234], "ASN 26210": [347], "ASN 21599": [196], "ASN 10906": [256], "ASN 6429": [34], "ASN 7738": [265], "ASN 6057": [109], "ASN 1916": [190], "ASN 27847": [162], "ASN 11830": [212], "ASN 7908": [46], "ASN 6503": [212], "ASN 23520": [181], "ASN 10429": [337], "ASN 27947": [212], "ASN 6568": [409], "ASN 6400": [190], "ASN 52482": [222], "ASN 14259": [143], "ASN 12066": [209], "ASN 10620": [203], "ASN 6471": [43], "ASN 23383": [172], "ASN 18678": [212], "ASN 52284": [197], "ASN 27755": [203], "ASN 11340": [396], "ASN 8167": [281], "ASN 15576": [184, 15, 284, 212, 249, 181, 206, 184, 253, 253, 268, 199, 147, 356, 246, 253, 284, 254, 18, 271, 268, 234, 209, 197, 237, 21, 62, 206, 253, 156, 231, 225, 24, 243, 256, 246, 178, 262, 159, 234, 215, 53, 209, 231, 562, 253, 206, 231, 278, 306, 184, 256, 200, 228, 375, 256, 68, 172, 184, 16, 290, 12, 18, 246, 240, 194, 231, 219], "ASN 3790": [265], "ASN 25620": [387, 543, 409], "ASN 8048": [212], "ASN 23201": [34, 102], "ASN 23243": [193], "ASN 21826": [175], "ASN 27887": [172], "ASN 12127": [243, 384], "ASN 19429": [262], "ASN 16629": [375]}}, "Medida 1": {"asn_origin ": "22927", "as_list": {...CONTINÚA...}}}

Continuando con la actividad, en lo referente a resultados de la relación entre *As-path* y latencia, se recurrió a los *traceroutes* contenidos en las medidas de CAIDA. Utilizando las herramientas desarrolladas, se crearon listas, dentro de las cuales se obtiene el camino IP que recorren los paquetes de la prueba correspondiente. Luego se interpreta este camino obteniendo los ASNs a los que pertenece cada IP del camino. De esta manera se obtiene, tanto los caminos entre ASNs, como los caminos dentro de un mismo ASN. Esto permite analizar el impacto de determinado camino IP, teniendo como resultado una aproximación al AS-Path.

A continuación se detalla un ejemplo de salida utilizando el script de interpretación en un *Shell* tipo UNIX. Las líneas punteadas separan distintas pruebas realizadas a determinado origen y destino:

IPs1
el camino es:
1. 27678 NIC Chile,CL
2. 6429 Telmex Chile Internet S.A.,CL
3. 6429 Telmex Chile Internet S.A.,CL
4. 6429 Telmex Chile Internet S.A.,CL

5. 2914 NTT-COMMUNICATIONS-2914 - NTT America, Inc.,US
6. 2914 NTT-COMMUNICATIONS-2914 - NTT America, Inc.,US
7. 1239 SPRINTLINK - Sprint,US
8. 1239 SPRINTLINK - Sprint,US
9. 7738 Telemar Norte Leste S.A.,BR
10. 7738 Telemar Norte Leste S.A.,BR
11. 7738 Telemar Norte Leste S.A.,BR
12. 7738 Telemar Norte Leste S.A.,BR
13. 7738 Telemar Norte Leste S.A.,BR
RTTs1
la cantidad de saltos es: 13
[0.133, 0.527, 1.02, 1.454, 107.919, 170.489, 171.35, 367.75, 410.721, 384.991, 406.621, 381.168, 407.058]
el RTT es: 407

IPs2
el camino es:
1. 6429 Telmex Chile Internet S.A.,CL
2. 6429 Telmex Chile Internet S.A.,CL
3. 6429 Telmex Chile Internet S.A.,CL
4. 2914 NTT-COMMUNICATIONS-2914 - NTT America, Inc.,US
5. 2914 NTT-COMMUNICATIONS-2914 - NTT America, Inc.,US
6. 701 UUNET - MCI Communications Services, Inc. d/b/a Verizon Business,US
7. 701 UUNET - MCI Communications Services, Inc. d/b/a Verizon Business,US
8. 701 UUNET - MCI Communications Services, Inc. d/b/a Verizon Business,US
9. 701 UUNET - MCI Communications Services, Inc. d/b/a Verizon Business,US
10. 15776 ASN no ANUNCIADO
11. 8151 Uninet S.A. de C.V.,MX
RTTs2
la cantidad de saltos es: 11
[0.618, 0.951, 1.331, 107.743, 107.919, 136.942, 172.759, 172.815, 236.352, 264.499, 280.058]
el RTT es: 280
IPs3
el camino es:
1. 27678 NIC Chile,CL
2. 6429 Telmex Chile Internet S.A.,CL
3. 6429 Telmex Chile Internet S.A.,CL
4. 6429 Telmex Chile Internet S.A.,CL
5. 2914 NTT-COMMUNICATIONS-2914 - NTT America, Inc.,US
6. 2914 NTT-COMMUNICATIONS-2914 - NTT America, Inc.,US
7. 701 UUNET - MCI Communications Services, Inc. d/b/a Verizon Business,US
8. 701 UUNET - MCI Communications Services, Inc. d/b/a Verizon Business,US
9. 701 UUNET - MCI Communications Services, Inc. d/b/a Verizon Business,US
10. 701 UUNET - MCI Communications Services, Inc. d/b/a Verizon Business,US
11. 15776 ASN no ANUNCIADO
12. 15776 ASN no ANUNCIADO
13. 15776 ASN no ANUNCIADO
14. 15776 ASN no ANUNCIADO

15. 15776 ASN no ANUNCIADO
16. 8151 Uninet S.A. de C.V.,MX
17. 8151 Uninet S.A. de C.V.,MX
RTTs3
la cantidad de saltos es: 17
[0.146, 0.539, 0.911, 1.164, 107.722, 107.649, 136.872, 230.039, 234.812, 211.216, 226.212, 228.185, 226.678, 227.402, 225.375, 223.796, 236.628]
el RTT es: 236
IPs1
el camino es:
1. 6429 Telmex Chile Internet S.A.,CL
2. 6429 Telmex Chile Internet S.A.,CL
3. 6429 Telmex Chile Internet S.A.,CL
4. 2914 NTT-COMMUNICATIONS-2914 - NTT America, Inc.,US
5. 2914 NTT-COMMUNICATIONS-2914 - NTT America, Inc.,US
6. 701 UUNET - MCI Communications Services, Inc. d/b/a Verizon Business,US
7. 701 UUNET - MCI Communications Services, Inc. d/b/a Verizon Business,US
8. 701 UUNET - MCI Communications Services, Inc. d/b/a Verizon Business,US
9. 701 UUNET - MCI Communications Services, Inc. d/b/a Verizon Business,US
10. 8151 Uninet S.A. de C.V.,MX
11. 8151 Uninet S.A. de C.V.,MX
RTTs1
la cantidad de saltos es: 11
[0.524, 0.884, 1.248, 107.771, 108.187, 137.08, 172.312, 172.32, 236.85, 264.307, 278.288]
el RTT es: 278
IPs4
el camino es:
1. 6429 Telmex Chile Internet S.A.,CL
2. 6429 Telmex Chile Internet S.A.,CL
3. 6429 Telmex Chile Internet S.A.,CL
4. 2914 NTT-COMMUNICATIONS-2914 - NTT America, Inc.,US
5. 2914 NTT-COMMUNICATIONS-2914 - NTT America, Inc.,US
6. 701 UUNET - MCI Communications Services, Inc. d/b/a Verizon Business,US
7. 701 UUNET - MCI Communications Services, Inc. d/b/a Verizon Business,US
8. 701 UUNET - MCI Communications Services, Inc. d/b/a Verizon Business,US
9. 701 UUNET - MCI Communications Services, Inc. d/b/a Verizon Business,US
10. 15776 ASN no ANUNCIADO
11. 15776 ASN no ANUNCIADO
12. 15776 ASN no ANUNCIADO
13. 15776 ASN no ANUNCIADO
14. 15776 ASN no ANUNCIADO
15. 8151 Uninet S.A. de C.V.,MX
16. 8151 Uninet S.A. de C.V.,MX
RTTs4
la cantidad de saltos es: 16
[0.545, 0.857, 1.266, 107.68, 107.814, 136.824, 204.02, 204.166, 206.387, 227.582, 227.216, 227.394, 226.085, 226.216, 223.286, 232.599]
el RTT es: 232

De esta manera se comparan los caminos correspondientes a una misma prueba.

Actividad 5 - Incorporación de Puntos de Intercambio de Tráfico y gestión de mediciones

La siguiente actividad buscó identificar el impacto que poseen la infraestructura de puntos de Intercambio de Tráfico en las redes de cada país.
Para llevar a cabo los desarrollos fue necesario consultar una base de datos con información sobre los *IXPs*.
Los administradores de esta base, *PeeringDB,* se encuentran desarrollando herramientas que permitirán obtener información acerca de los IXPs de manera mucho más accesible, pero estas herramientas no están disponibles a la finalización de esta Contribución, por lo que se optó por realizar un desarrollo que involucre solamente un país de manera de optimizar tiempo, ya que la metodología de obtención de datos actual, es de manera manual, lo que insume un tiempo demasiado grande.
A su vez, para integrar los desarrollos realizados es necesario hacer modificaciones a la base de datos central de Simón, las cuales también se están llevando a cabo por el *staff de LACNIC*. En este sentido los scripts desarrollados como resultado de la contribución, ya poseen todas las capacidades para que su integración no implique nuevas modificaciones luego de que la base de datos de Simón esté completamente modificada.
Finalmente y con las metodologías explicadas en el desarrollo se realiza un relevamiento de los resultados a través del script *ixp_comparator.py*[], obteniéndose los siguientes valores para el IXP de Ecuador “NAP.ec”:

- RTT mediana dentro del IXP 22.000000 ms
  Cantidad de medidas dentro del IXP:123

- RTT mediana fuera del IXP 14.500000 ms
  Cantidad de medidas fuera del IXP:30

Como resultado del mecanismo y scripts elaborados, es ahora posible evaluar (por ahora en forma limitada, hasta que PeeringDB complete la API que permita adquirir los datos completos de IXPs la región) el impacto de, en principio, cualquier IXP en un determinado país, en base a los datos almacenados en la base de Simón

Actividad 6 - Modificación a plataformas actuales

Otra actividad proyectada fue la modificación de las herramientas de Simón para gestionar el tráfico de distintas maneras. A comienzo del proyecto se pretendía seguir una línea de trabajo que permitiera al medidor, realizar pruebas a múltiples destinos distintos, aparte de los servidores de *SpeedTest*, pero esta idea fue luego desestimada. Se plantea entonces continuar utilizando los servidores de *SpeedTest*, pero siguiendo algoritmos más específicos a la hora de elegir contra que servidor hacer la prueba de latencia, en lugar de hacerlo de manera aleatoria
Con este fin se modificó un script ya existente, en forma de función de *Python*, para ser integrado más rápidamente al proyecto, en el cual se establecen 3 lineamientos para hacer las pruebas:

- Una prueba debe ser realizada al servidor de *SpeedTest* del país de origen. En caso de no haberlo se utiliza uno aleatoriamente.
- Una segunda prueba debe ser realizada al país de destino, que tengas menos medidas desde el país de origen
- El resto de las pruebas se realizan a servidores de manera aleatoria.

A partir de estos lineamientos resultó el correcto desarrollo de la mencionada función de Python, cumpliendo con los requisitos establecidos.

A modo de ejemplo se detalla la salida de la función, donde se solicitan 5 puntos, con destinos IPv4 con origen de pruebas desde Uruguay. Como se puede apreciar resaltado se cumplen todos los requerimientos establecidos

>>> testpoint_manage(5,4,'UY') [solicito 5 puntos, IPv4 para pruebas desde UY]

[Lista de medidas de Uruguay] = {u'DO': 882, u'GT': 730, u'CO': 1118, u'VE': 624, u'CL': 1404, u'BO': 943, u'HT': 212, u'AS': 329, u'AR': 2996, u'HN': 557, u'EC': 762, u'BR': 14200, u'EU': 240, u'CR': 714, u'CU': 127, u'PR': 138, u'NI': 421, u'UY': 294, u'SR': 139, u'TT': 222, u'PY': 925, u'SV': 693, u'US': 670, 'cc_base': 'UY', u'PA': 1077, u'BZ': 343, u'PE': 1008, u'MX': 8514}

[País con menor cantidad de medidas] = CU

[Punto de test del país de origen] = [<TestPoint: 200.40.30.8>]

'{ "points": [{"city": "La Habana", "url": "http://www.trabajadores.cu", "ip": "190.92.127.20", "region": 29, "countryName": "Cuba", "country": "CU", "images": "[]"}, {"city": "Montevideo", "url": "speedtest.com.uy", "ip": "200.40.30.8", "region": 5, "countryName": "Uruguay", "country": "UY", "images": "[{\\"name\\": \\"random350x350.jpg\\", \\"height\\": 350, \\"width\\": 350, \\"timeout\\": 4000, \\"path\\": \\"/speedtest/random350x350.jpg\\", \\"type\\": \\"jpg\\", \\"size\\": 245388}, {\\"name\\": \\"random500x500.jpg\\", \\"height\\": 500, \\"width\\": 500, \\"timeout\\": 8000, \\"path\\": \\"/speedtest/random500x500.jpg\\", \\"type\\": \\"jpg\\", \\"size\\": 505544}, {\\"name\\": \\"random750x750.jpg\\", \\"height\\": 750, \\"width\\": 750,

\\"timeout\\": 16000, \\"path\\": \\"/speedtest/random750x750.jpg\\", \\"type\\": \\"jpg\\", \\"size\\": 1118012}]"}, {"city": "Valinhos", "url": "campinas.speedtest.acessoturbo.net.br", "ip": "186.209.39.23", "region": 5, "countryName": "Brazil", "country": "BR", "images": "[{\\"name\\": ……..[CONTINUA]

Actividad 7 - Índice de conectividad

Una de las actividades finales proyectadas, fue el estudio de un índice de conectividad, el cual establece un indicador de calidad que posee un país, un ASN en particular o una región en cuanto a conectividad.
Al inicio del proyecto se previó fabricar este índice en base a mediciones de latencias entre países, para luego establecer una escala de calidad según este índice. Luego de realizar un análisis más exhaustivo de la posible solución, se definieron como resultado más variables, que inciden mejorando la calidad de dicho índice de conectividad regional.

Otras actividades mencionadas en el anteproyecto

Si bien en el anteproyecto se planificó la realización de modificaciones de visualización al medidor de *Javascript*, posteriormente se decidió no llevar a cabo esta actividad de naturaleza "cosmética", en pro de invertir el tiempo disponible en las actividades tendientes a los resultados antes mencionados.

Capítulo 9 – Conclusiones

Durante el desarrollo de este capítulo de detallan las conclusiones de cada una de las actividades realizadas.

Conclusiones actividad 1 - Herramientas de trabajo

Teniendo en cuenta las etapas planificadas, las mismas se completaron de manera muy satisfactoria, siendo la base para sucesivos eventuales desarrollos y mejoras al proyecto Simón.

Conclusiones actividad 2 - Normalización de datos

Teniendo en cuenta los resultados obtenidos en la normalización de los datos según el par sistema operativo navegador, a pesar de que varias de las mismas poseen un factor de corrección similar a la desviación estándar calculada, los porcentajes de corrimiento se pueden considerar notorios.
Esto permite concluir que efectivamente las medidas pueden verse afectadas según el tipo de sistema operativo o navegador que se utilice, por lo que la normalización de los datos se vuelve imprescindible a la hora presentar y comparar los resultados.

Como fue mencionado en la sección de resultados, para el caso de la normalización de medidas en función de cada medidor, la misma decide no realizarse, donde se tuvo en cuenta como factores principales para tomar la decisión, la utilización de los medidores y la facilidad de implementación.
En el caso de TCP y NTP, son los medidores con los cuales se comenzó el proyecto Simón, pero desde el año 2012 que no se obtienen resultados de los mismos. Dado esto, desde el staff de LACNIC se apunta a obtener mayor cantidad de información con los demás medidores contenidos en el proyecto (*JavaScript y partners)* y disminuir el uso de los medidores TCP y NTP, de manera que no se utilicen en el futuro. A su vez las facilidades que provee el medidor de *JavaScript*, hacen que sea más simple de implementar y de utilizar por parte de los usuarios, ya que solamente implica ingresar a la página web del proyecto Simón. La opción de alojar el código *JavaScript* en una página web particular (distinta a la del proyecto Simón), es también más sencilla de aplicar que la integración de los medidores TCP y NTP.
En lo que respecta a RIPE y CAIDA, ya que utilizan ICMP como protocolo de pruebas, en lugar de normalizar los datos de estos, con los obtenidos por el medidor *Javascript,* se concluye que es más beneficioso tener una comparativa entre las mediciones de latencia utilizando ICMP y las mediciones de respuesta HTTP, buscando identificar distintos comportamientos en las redes.
Por lo tanto, al día de hoy, no se considera que la capa de normalización de los datos de distintos medidores sea beneficiosa, pero el estudio realizado de cada uno de los medidores permite revisar esta decisión más rápidamente en el futuro, permitiendo

también estandarizar el proceso de comparación, en caso de que surjan nuevos métodos de medición.

Conclusiones actividad 3 - Incorporación de mediciones de *partners* internacionales

En lo que respecto a la obtención de medidas de partners internacionales, teniendo en cuenta primeramente a RIPE, a pesar de que el volumen de información fue menor al esperado, la estructura y la disposición de las *probes*, en la mayoría de los casos colocadas detrás de un servicio residencial, con un arquitectura distribuida en la región, hace que las medidas de latencia obtenidas por este medio, reflejen en mayor medida la calidad en la conectividad de los usuarios finales. En cambio la disposición estratégica de los medidores de CAIDA, dentro de redes muy robustas y elegidas de manera estratégica, en general dentro de ISPs o IXPs, hace que los resultados obtenidos sean representativos de una parte de la región más acotada y con medidas que no necesariamente reflejan la calidad de la red que se provee al usuario final.
Se puede concluir por lo tanto, que basados en los datos de RIPE se obtienen resultados de medidores mayormente distribuidos en la región y que reflejan la calidad de los enlaces en el extremo final del cliente. A su vez, el proyecto Simón ya se encuentra preparado para adoptar las medidas de este *partner* en caso que la cantidad de medidas o de *probes* aumenten en la región, lo cual es un objetivo de LACNIC como organización, ya que incentiva la colocación de las mismas.
En base a los datos de CAIDA, se obtiene un flujo muchísimo más grande de información, que puede ser aprovechada para obtener conclusiones complementarias, al comparar datos entre resultados de un medidor u otro, teniendo en cuenta no solo las medidas de latencia, sino también los caminos de *traceroutes* obtenidos.
Por lo tanto la inclusión de las medidas de ambos *partners* enriquece la base de datos de Simón en cantidad por el lado de CAIDA y en calidad de puntos de prueba por el lado de RIPE.

Conclusiones actividad 4 - Incorporación de sistemas autónomos

En lo que respecta a la matriz de latencias y ASNs, utilizando las herramientas de Simón se logró procesar mayor cantidad de información para generar la matriz deseada, por lo que a pesar de generarse de manera incompleta en el ambiente de maqueta, puede ser culminada en su totalidad en un servidor con capacidades de procesamiento mayores, obteniéndose así una nueva forma de comparar las medidas de latencia, no solo entre destinos IP, sino también entre ASNs

Por el lado de la Relación AS-path y latencia, teniendo en cuenta las muestras de información tomadas de CAIDA utilizando el script *as_path_reader.py*[], con los 3 ciclos de prueba de 48 horas, se observa que:

- Existen casos donde el ASN destino es el mismo, pero la diferencia en los caminos dentro de un mismo ASN, hace que las latencias varíen, donde no necesariamente el camino más largo es el más lento.
- Una gran cantidad de resultados poseen saltos en ASNs que corresponden a empresas fuera de la región de LACNIC, siendo la prueba de un origen de LACNIC a un destino en LACNIC. Esta situación es conocida dentro de la región, dado que existen empresas de gran porte en América Latina y el Caribe que poseen enlaces con el NAP de las Américas (https://es.wikipedia.org/wiki/NAP_de_las_Am%C3%A9ricas) por lo que es interesante también relevar la información de cuáles son los resultados en caso que el trafico sea contenido dentro de la región.

Teniendo en cuenta estas observaciones y a pesar de que se trabajó solamente 3 ciclos se concluye que existe una relación de dependencia entre las medidas de latencia y el *as-path,* no solo dada por la cantidad de AS por los cuales se transita, sino también si la información circula solamente dentro de la región de LACNIC, o tiene saltos hacia afuera de la misma. En el futuro con esta herramienta y más ciclos de pruebas de CAIDA se puede obtener un mapa más detallado de la situación regional.

Conclusiones actividad 5 - Incorporación puntos de intercambio de trafico

Teniendo en cuenta la información dispuesta en el capítulo de resultados, los valores de latencia media obtenidos dentro del *IXP* son mayores a los obtenidos por fuera del mismo. Esta situación no es la esperada, pero puede explicarse teniendo en cuenta que la cantidad de medidas disponible dentro del punto de intercambio de tráfico utilizado como ejemplo cuadriplican a las medidas obtenidas por fuera del mismo; a la vez que la ubicación geográfica del país referido puede no presentar una diferencia sustancial respecto a caminos entre ISPs del mismo país por fuera del IXP. De cualquier manera, las diferencias notorias entre la cantidad de medidas obtenidas dentro y fuera del IXP, permiten confirmar que el *mismo* fomenta el intercambio de tráfico interno en un país o en la región; y el sistema desarrollado permite la realización de este tipo de estadísticas para cualquier IXP de la región del que se disponga de datos.

Conclusiones actividad 6 - Modificación a plataformas actuales

Teniendo en cuenta los lineamientos descritos en la sección de resultados, elegidos en base a necesidades actuales del proyecto, se concluye que las modificaciones satisfacen la necesidad del proyecto en cuanto a diversificación de medidas, aportando valor a los nuevos resultados obtenidos a partir de ellas, siendo el nuevo entregable realizado lo suficientemente versátil para modificar estos lineamientos establecidos de manera ágil en caso de ser necesario.

Conclusiones actividad 7 - Índice de conectividad

Las variables mencionadas en el trascurso de esta actividad hacen que la generación del índice no sea trivial, debiendo tener en cuenta una multiplicidad de factores y no solamente medidas de latencia.
A su vez se plateó el impacto que puede tener un índice, que no comprenda múltiples variables, ya que desde la posición de LACNIC, definir un índice, puede generar riesgos de relacionamiento, entre organizaciones o países involucrados.
Se concluye entonces, teniendo en cuenta los objetivos planteados que el estudio de un posible índice y los factores que lo componen ha sido completado.
Luego en futuras contribuciones, se puede definir el peso de cada una de las variables y los datos que las respaldan, pudiendo finalmente definir un índice que represente de aún mejor la calidad de la conectividad en América Latina y el Caribe.

Resumen final de conclusiones

Resumiendo lo mencionado en esta sección, se concluye que los objetivos planteados inicialmente fueron alcanzados, algunos de los cuales implicaron replanteos y toma de decisiones, que llevaron a adecuar algunas de las acciones y caminos planificados al comienzo. A su vez esta contribución posibilita diferentes ramificaciones de trabajo futuros, lo que permite seguir avanzando en la ampliación y mejoramiento de las herramientas existentes y desarrolladas.

Referencias

[1] Proyecto Simón (2015, Julio 30) [Online]. Disponible: http://simon.lacnic.net/simon

[2] Position Paper - Alternatives to traditional latency measurement techniquesn (2014, Junio) [Online]. Disponible: http://simon.lacnic.net/simon/static/simon_app/articles/PositionPaper-Alternativestotraditionallatencymeasurementtechniques.pdf

[3] Objetivos Proyecto Simón (2015, Julio 30) [Online]. Disponible: http://simon.lacnic.net/simon/objectives/

[4] Wikipedia, Representation State Tranfers (2015, Julio 30) [Online]. Disponible: http://en.wikipedia.org/wiki/Representational_state_transfer

[5] Wikipedia, Application programming interface (2015, Julio 30) [Online]. Disponible: http://en.wikipedia.org/wiki/Application_programming_interface

[6] Wikipedia, JavaScript Object Notation (2015, Agosto 1) [Online]. Disponible: http://en.wikipedia.org/wiki/JSON

[7] Wikipedia, RTT (2015, Junio 28) [Online]. Disponible: https://en.wikipedia.org/wiki/Round-trip_delay_time

[8] Python (2015, Julio) [Online]. Disponible: http://www.python.org

[9]Django Project (2015, Julio) [Online] Disponible: https://www.djangoproject.com/

[10] Django API *models* versión 1.6 (2015, Julio) [Online] Disponible: https://docs.djangoproject.com/en/1.6/ref/models/instances/#django.db.models.Model.save)

[11] Django API *models* versión 1.8 (2015, Julio) [Online] Disponible: https://docs.djangoproject.com/en/1.8/topics/db/models/

[12] Pycharm (2015, Julio) [Online] Disponible: https://www.jetbrains.com/pycharm/

[13] Librería Python-dev (2015, Julio) [Online] Disponible: https://packages.debian.org/wheezy/python-dev

[14] Librería Libpq-dev (2015, Julio) [Online] Disponible: https://packages.debian.org/es/sid/libpq-dev

[15] Adaptador de consultas multi-hilos Psycopg-2 (2015, Junio 29) [Online] Disponible: https://pypi.python.org/pypi/psycopg2

[16] Paquete de manipulación de direcciones IP Netaddr (2015, Junio 29) [Online] Disponible: https://pypi.python.org/pypi/netaddr

[17] Adaptador de XML para *Python* (2015, Julio) [Online] Disponible: https://pypi.python.org/pypi/ixml/0.1.0

[18] *About Vagrant* (2015, Julio) [Online] Disponible: https://www.vagrantup.com/about.html

[19]*Vagrant Documentation* (2015, Julio) [Online] Disponible: https://docs.vagrantup.com/v2/

[20] *VirualBox* (2015, Julio) [Online] Disponible: https://www.virtualbox.org/

[21]. *Vagrant, Synced Folders* (2015, Julio) [Online] Disponible: https://docs.vagrantup.com/v2/getting-started/synced_folders.html

[22] *Vagrant, Forwarded Ports* (2015, Julio) [Online] Disponible: http://docs.vagrantup.com/v2/networking/forwarded_ports.html

[23] Wikipedia, TCP (2015, Mayo 5) [Online] Disponible: https://es.wikipedia.org/wiki/Transmission_Control_Protocol

[24] Wikipedia, NTP (2015, Abril 1) [Online] Disponible:
https://es.wikipedia.org/wiki/Network_Time_Protocol

[25] Wikipedia, *JQuery* (2015, Junio 6) [Online] Disponible:
https://es.wikipedia.org/wiki/JQuery

[26] Como participar, Proyecto Simón (2015, Julio) [Online] Disponible:
http://simon.lacnic.net/simon/participate/

[27] Browserstack (2015, Julio) [Online] Disponible:
www.browserstack.com

[28] StatCounter (2015, Julio) [Online] Disponible:
http://gs.statcounter.com/

[29] Netmarketshare (2015, Julio) [Online] Disponible:
http://marketshare.hitslink.com/

[30] Sistemas operativos Netmarketshare periodo 2013 -2015 (2015, Mayo) [Online] Disponible:
http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0&qpsp=2013&qpnp=3&qptimeframe=Y

[31] Sistemas operativos, StatCounter periodo 2013 -2015 (2015, Mayo) [Online] Disponible:
http://gs.statcounter.com/#desktop-os-sa-monthly-201301-201503

[32] Navegadores, Netmarketshare periodo 2013 -2015 (2015, Mayo) [Online] Disponible:
http://marketshare.hitslink.com/browser-market-share.aspx?qprid=0&qpcustomd=0&qpsp=2013&qpnp=3&qptimeframe=Y

[33] Navegadores, StatCounter periodo 2013 -2015 (2015, Mayo) [Online] Disponible:
http://gs.statcounter.com/#desktop-browser-sa-monthly-201101-201503

[34] FAQ, StatCouter (2015, Mayo) [Online] Disponible:
http://gs.statcounter.com/faq#methodology)

[35] FAQ, Netmarketshare (2015, Mayo) [Online] Disponible: (http://www.netmarketshare.com/faq.aspx)

[36] simon_probe_plugin.js (2015, Julio) [Local] Disponible: \Contribucion_ proyecto_Simon\Scripts\Capitulo_2

[37] Browserstack, Python, Configuring Capabilities (2015, Julio) [Online] Disponible: https://www.browserstack.com/automate/python

[38] browsersOSsTests.py (2015, Julio) [Local] Disponible: \Contribucion_ proyecto_Simon\Scripts\Capitulo_2

[39] Histograma_juan.py (2015, Julio) [Local] Disponible: \Contribucion_ proyecto_Simon\Scripts\Capitulo_2

[40] RIPE NCC (2015, Julio) [Online] Disponible: https://www.ripe.net/

[41] CAIDA (2015, Julio) [Online] Disponible: http://www.caida.org/home/

[42] LACNIC (2015, Julio) [Online] Disponible: http://www.lacnic.net/web/lacnic/ipv4

[43] RIPE Atlas project (2015, Julio) [Online] Disponible: https://atlas.ripe.net/about/.

[44]Probe (2015, Julio) [Online] Disponible: https://labs.ripe.net/Members/suzanne_taylor_muzzin/ripe-atlas-april-2013-update

[45]Network coverage, RIPE Atlas probes (2015, Julio) [Online] Disponible: https://atlas.ripe.net/results/maps/network-coverage/

[46] Network coverage, RIPE Atlas anchors (2015, Julio) [Online] Disponible: https://atlas.ripe.net/anchors/map/

[47] RIPE Community (2015, Julio) [Online] Disponible:
https://atlas.ripe.net/get-involved/community/

[48] RIPE Altas REST Api (2015, Julio) [Online] Disponible:
https://atlas.ripe.net/docs/rest/

[49] retrieve_ripe_db.py (2015, Julio) [Local] Disponible:
\Contribucion_ proyecto_Simon\Scripts\Capitulo_3

[50] atlas_DB_results.py (2015, Julio) [Local] Disponible:
\Contribucion_ proyecto_Simon\Scripts\Capitulo_3

[51] DB_creator.py (2015, Julio) [Local] Disponible:
\Contribucion_ proyecto_Simon\Scripts\Capitulo_3

[52] CAIDA, Archipielago ARK project (2015, Julio) [Online] Disponible:
http://www.caida.org/projects/ark/

[53] CAIDA monitor (2015, Julio) [Online] Disponible:
http://www.caida.org/projects/ark/ark-raspi-monitor-20130102.png

[54] Archipielago Monitors Locations (2015, Julio) [Online] Disponible:
http://www.caida.org/projects/ark/locations/

[55] Route Views (2015, Julio) [Online] Disponible:
http://www.routeviews.org/.

[56] The IPv4 Router /24 Topology Datasets (2015, Julio) [Online] Disponible:
http://www.caida.org/data/active/ipv4_routed_24_topology_dataset.xml

[57] Paris Traceroute (2015, Julio) [Online] Disponible:
http://www.paris-traceroute.net/about

[58] Estadísticas medidor scl-cl, NIC Chile (2015, Julio) [Online] Disponible:
http://www.caida.org/projects/ark/statistics/scl-cl/ip_dispersion.html

[59] Probe Data (2015, Julio) [Online] Disponible: http://data.caida.org/datasets/topology/ark/ipv4/

[60] Scamper (2015, Julio) [Online] Disponible: http://www.caida.org/tools/measurement/scamper/

[61] caída-fetch (2014, Julio) [Local] Disponible: ANEXO

[62] caida_DB_creator.py (2015, Julio) [Local] Disponible: \Contribucion_ proyecto_Simon\Scripts\Capitulo_3

[63] Wikipedia, ASN (2015, Junio 13) [Online] Disponible: https://es.wikipedia.org/wiki/Sistema_aut%C3%B3nomo

[64] asnMatrixCreator.py (2015, Julio) [Local] Disponible: \Contribucion_ proyecto_Simon\Scripts\Capitulo_4

[65] asnMatrixRIPEretriever.py (2015, Julio) [Local] Disponible: \Contribucion_ proyecto_Simon\Scripts\Capitulo_4

[66] RIPE, RIS (2015, Julio) [Online] Disponible: https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris

[67] scamper_complete_file.py (2015, Julio) [Local] Disponible: \Contribucion_ proyecto_Simon\Scripts\Capitulo_4

[68] caidaAsPath.py (2015, Julio) [Local] Disponible: \Contribucion_ proyecto_Simon\Scripts\Capitulo_4

[69] caidaAsPathNew.py (2015, Julio) [Local] Disponible: \Contribucion_ proyecto_Simon\Scripts\Capitulo_4

[70] as_path_reader.py (2015, Julio) [Local] Disponible: \Contribucion_ proyecto_Simon\Scripts\Capitulo_4

[71] ISOC, Puntos de Intercambio de tráfico (IXP), Documentos informativos (2015, Julio) [Online] Disponible: https://www.internetsociety.org/sites/default/files/Internet%20Exchange%20Points%20-%20Spanish.pdf

[72] PeeringDB (2015, Julio) [Online] Disponible: https://www.peeringdb.com

[73] ixp_measure.py (2015, Julio) [Local] Disponible: \Contribucion_ proyecto_Simon\Scripts\Capitulo_5

[74] PeeringDB, versión beta (2015, Julio) [Online] Disponible: https://beta.peeringdb.com

[75] ixp_comparator.py (2015, Julio) [Local] Disponible: \Contribucion_ proyecto_Simon\Scripts\Capitulo_5

[76] testpoint_manage.py (2015, Julio) [Local] Disponible: \Contribucion_ proyecto_Simon\Scripts\Capitulo_6

ANEXO

caída-fetch – Script para obtención de archivos WARTS

#!/bin/bash

```
BASE_URL=http://data.caida.org/datasets/topology/ark/ipv4/probe-data
CURRENT_YEAR=$(date +"%Y")
TEAMS="team-1 team-2 team-3"
YEARS=$(seq $CURRENT_YEAR 2008)
YEARS=$(seq 2009 2008)

# case $1 in
#             'all')          echo "Fetching all data..."
#                             wget -q -r -np -nd -A *mty-mxx*,*sao-br*,*sao2-br*,*gig-br*,*scl-cl*,*scl2-
cl*,*kna-cl*,*sju-pr* $BASE_URL
#                             exit 0
#             ;;
#             'fast')
#                             for team in $TEAMS
#                             do
#                                         for year in $YEARS
#                                         do
#                                                     URLs=$URLs $BASE_URL/$team/$year/
#                                         done
#
#                             done
#
#                             echo $URLs | xargs -n 1 -P 8 wget -q -r -np -nd -A *mty-mxx*,*sao-br*,*sao2-
br*,*gig-br*,*scl-cl*,*scl2-cl*,*kna-cl*,*sju-pr*
#                             exit 0
#             ;;
# esac


for year in $YEARS
do
            for server in sao-br sao2-br gig-br scl-cl scl2-cl kna-cl sju-pr mty-mx
            do
                        dir=$server$year
                        [ -d $dir ] && echo "Skipping $dir - already scanned" && continue
                        mkdir $dir
                        echo "Fetching $dir"

                        for team in $TEAMS
                        do
                                    wget -q --directory-prefix=$dir -r -np -nd -A *$server*
$BASE_URL/$team/$year/
                        done
            done
done
```