
Proyecto de Grado

Presentado ante la ilustre Universidad de Los Andes como requisito parcial para obtener el Título de Ingeniero de Sistemas

# Diseño e implementación de un sistema de monitoreo de redes orientado a la recolección masiva de datos.

Por

Br. Jesús Alberto Gómez Pérez

Tutor: Dr. Andrés Arcia-Moret

Noviembre 2015

# Diseño e implementación de un sistema de monitoreo de redes orientado a la recolección masiva de datos.

Br. Jesús Alberto Gómez Pérez

Proyecto de Grado — Sistemas Computacionales, 58 páginas

**Resumen:** En el presente proyecto se plantea el desarrollo de un sistema de monitoreo distribuido de enlaces críticos de redes a través de un servicio web centralizado. Este servicio además pretende hacer énfasis en la visualización de los datos recolectados a partir de pruebas periódicas. El sistema está planteado como una herramienta para facilitar el entendimiento del funcionamiento de la red y ofrecer una solución centralizada, de muy bajo costo y con componentes de hardware que puedan estar desatendidos.

**Palabras clave:** Monitoreo de redes, Calidad de servicio, Big Data, Benchmarking, Cloud

# Índice

# Capítulo 1

# Introducción

## 1.1 Antecedentes

Existen dos estrategias de monitoreo de redes: monitoreo activo o benchmarking que consiste en generar tráfico para realizar medidas y comprobar la respuesta de la red y monitoreo pasivo que consiste en escanear el tráfico de la red en ciertos puntos estratégicos para censar el tráfico en la red. El benchmarking tiene la desventaja de tener que inyectar tráfico lo cual puede entorpecer el funcionamiento normal de la red ejemplos de herramientas de benchmarking son ping, iperf y traceroute.

El monitoreo pasivo tiene la ventaja de que nos puede dar una muy buena idea del uso de la red sin embargo para mayor efectividad debe realizarse en nodos intermedios a los que muchas veces no tenemos acceso, ejemplos de herramientas de monitoreo pasivo son tcpdump y wireshark; en ambos casos hay que resaltar que es difícil tener una imagen completa de la realidad de la red.

Uno de los trabajos más importantes en el área de monitoreo de redes es el Protocolo Simple de Administración de Red o SNMP (del inglés Simple Network Management Protocol) este emergió como una de las primeras soluciones al problema de manejo de redes y se ha convertido en la solución más ampliamente aceptada debido a su diseño modular e independiente de productos o redes específicas. SNMP consiste de (1) un administrador de red, (2) una serie de dispositivos remotos monitoreados (3) bases de información de administración (MIBs) en estos dispositivos, (4) agentes remotos que

reportan la información de las MIBs al administrador de red y toman acciones si les indica y (5) un protocolo de comunicación entre los dispositivos [1].

SNMP no solo ofrece al administrador de red reportes sobre cada uno de los dispositivos administrados sino que también permite tomar acción sobre ellos proactivamente antes de que ocurran problemas o de forma reactiva para solucionar problemas cuando ocurren de forma inesperada.

La red de TVWS (TV White Spaces o Espacios blancos en el espectro radioeléctrico) de Malawi fue implementada en el marco de un proyecto para llevar Internet a áreas rurales en países en vías de desarrollo utilizando soluciones de bajo costo a través de los espacios en blanco en el espectro radioeléctrico específicamente en la banda UHF (siglas del inglés Ultra High Frequency, 'frecuencia ultra alta') [2].

Muchas veces esta red debe dejarse desatendida durante largos periodos de tiempo ya que sus nodos son de difícil acceso o es muy costoso tener a profesionales dedicados que se encarguen de su mantenimiento, este escenario hace evidente la necesidad de una solución de monitoreo de redes a distancia y de mínimo mantenimiento. Además es atractivo para el centro de monitoreo de la red poder añadir, modificar o eliminar nodos de interés de manera sencilla a la interfaz de monitoreo.

Para intentar solucionar este problema y recolectar información sobre la red de Malawi se implementó un sistema en dos partes: un monitor de red instalado en la estación base (BS) en Malawi y un servicio web remoto, el monitor en la estación base se conecta a cada uno de los nodos de la red y determina el tiempo de ida y vuelta (RTT por sus siglas en ingles) de forma automatizada en ciertos intervalos de tiempo, guarda los resultados en archivos y los coloca en una carpeta que se sincroniza a través de un servicio en la nube de tipo PaaS (Plataforma como servicio) con el servidor web, que a su vez escanea la carpeta compartida y actualiza su base de datos que puede usarse para generar gráficas de RTT promedio y determinar tiempos de actividad continuos y porcentaje de disponibilidad de servicio [3].

A pesar de que este sistema recolecta información útil y presenta gráficas muy sencillas de entender, su programación no permite agregar nuevos nodos, esto trae como consecuencia que debe ser modificado manualmente cuando la red se expande, dando lugar a la necesidad de que el servicio web se pueda expandir para dar servicio

a múltiples monitores remotos simultáneamente.

Por estos motivos proponemos la construcción de un sistema de monitoreo a gran escala que llamaremos Octopus Monitor, para hacerlo totalmente configurable y robusto además de agregar una interfaz de configuración vía web que permita manejar usuarios, agregar monitores remotos, agregar nodos y modificar los parámetros de las pruebas, todo esto apoyándonos en un sistema de archivos compartidos a través de la nube.

Mientras que el mayor valor de Malawinet Monitor es la visualización de grandes volúmenes de datos que se pueden obtener a partir de pruebas de bajo impacto de tráfico (ping, traceroute), se han realizado otros trabajos en el área de monitoreo de redes como Bowlmap; este es un sistema de monitoreo de redes a través de la visualización de mediciones para el Laboratorio Abierto Inalámbrico de Berlín (BOWL, por sus siglas en ingles). Este sistema tiene una alta flexibilidad ya que permite realizar cambios en sus pruebas existentes así como agregar pruebas totalmente nuevas y a su vez generar las visualizaciones necesarias para el análisis de dicha información, además tiene la ventaja de solo transmitir la información necesaria para cada actualización lo que acelera las peticiones y permite la visualización de data en tiempo real [4].

## 1.2 Planteamiento del Problema

Las redes de computadoras se componen de un conjunto de nodos interconectados sujetos a numerosos factores que escapan de nuestro control y sobre los que muchas veces no tenemos conocimiento, en otras palabras la red puede llegar a ser impredecible y no ofrece garantías sobre el servicio que ofrece; algunas aplicaciones dependen de una alta disponibilidad y estabilidad por lo que es esencial para un administrador de red tener información del estado de la red, para diagnosticar, solucionar problemas y asegurar la calidad de servicio.

Existen muchos otros ejemplos en los que es importante tener datos del estado de la red como en redes de bajo costo en las que pueden ocurrir largas interrupciones de servicio o para un cliente de un servicio de alojamiento web que desea saber si su sitio web está disponible y que tan rápido responde; plataformas como Pingdom [5] o

UptimeRobot [6] permiten monitorear distintos servicios en Internet y generan alertas cuando encuentran problemas, sin embargo no son gratuitas y no permiten la inclusión de nuevos tipos de pruebas o la visualización masiva de datos históricos.

Mantener estas mediciones con las herramientas existentes se vuelve una tarea compleja mientras crece el número de nodos a monitorear (es decir, las fuentes de información) y la cantidad de datos aumenta a través del tiempo, sumado a esto solo podemos capturar información a partir de los nodos externos de la red, por lo que en la mayoría de los casos no es posible tener una imagen completa de los enlaces a monitorear.

A pesar de que Malawinet Network Monitor podría ofrecer estadísticas de tiempo de ida y vuelta (RTT) y disponibilidad de un enlace solo a partir de las trazas capturadas con Ping, se desea además implementar un marco de trabajo que permita agregar nuevas pruebas automatizadas que ayuden a obtener una imagen más completa de la red.

## 1.3 Justificacion

Ya que las redes están compuestas de arreglos de nodos conectados entre si y muchas veces resultan de despliegues caóticos y no-coordinados sobre los que no se tiene control o conocimiento y existe una enorme cantidad de variables como

## 1.4 Objetivos

### 1.4.1 Objetivos Generales

Construir un servicio web de monitoreo de redes de bajo costo para países en vías de desarrollo con almacenamiento de datos en la nube de tipo PaaS que sea de fácil instalación y permita configurar múltiples monitores remotos para ajustarse a cambios en las características de las redes a monitorear y la carencia de personal in sitio.

### 1.4.2 Objetivos Específicos

- Desarrollar un servicio de monitoreo de bajo costo para países en vías de desarrollo que de servicio a múltiples monitores de red remotos, presente visualizaciones gráficas a partir de los datos recogidos y ofrezca un marco de trabajo para agregar nuestros tipos de pruebas a los monitores de red existentes.
- Desarrollar un cliente monitor para desplegar en nodos desatendidos con dispositivos recolectores de muestra de bajo costo (ej. Raspberry PI, Alix boards, APU) para observar el comportamiento de los enlaces a través de aplicaciones de monitoreo sencillas y de consola.
- Utilizar de sistemas de bajo costo y alta disponibilidad en la nube para almacenamiento y transferencia de datos.
- Integrar los distintos subsistemas que conforman el servicio de monitoreo.
- Desarrollar un modulo de calculo asíncrono de gráficas que permita mejorar los tiempos de interacción del usuario final con el sistema utilizando técnicas para agilizar cómputo como caching, prefetching, threads, etc.

## 1.5 Metodología

En este trabajo se seguirá una metodología en espiral; el modelo en espiral es un modelo del ciclo de vida del software donde el esfuerzo del desarrollo es iterativo. Cada ciclo de la espiral representa una fase del desarrollo de software, cada uno de los ciclos consiste de los siguientes pasos:

1. Determinar o fijar los objetivos. En este paso se definen los objetivos específicos para posteriormente identifica las limitaciones del proceso y del sistema de software, además se diseña una planificación detallada de gestión y se identifican los riesgos.
2. Análisis del riesgo. En este paso se efectúa un análisis detallado para cada uno de los riesgos identificados del proyecto, se definen los pasos a seguir para reducir los riesgos y luego del análisis de estos riesgos se planean estrategias alternativas.

3. Desarrollar, verificar y validar. En este tercer paso, después del análisis de riesgo, se eligen un paradigma para el desarrollo del sistema de software y se lo desarrolla.

4. Planificar. En este último paso es donde el proyecto se revisa y se toma la decisión si se debe continuar con un ciclo posterior al de la espiral. Si se decide continuar, se desarrollan los planes para la siguiente fase del proyecto.

Se realizarán cuatro ciclos, el primero corresponde a la realización de un monitor remoto básico que realice mediciones de RTT de la red con almacenamiento en la nube y visualizaciones de los datos obtenidos.

El segundo ciclo consiste en permitir el monitoreo de una cantidad arbitraria de monitores remotos permitiendo a múltiples usuarios manejar sus monitores remotos desde el servicio web y observar las visualizaciones.

El tercer ciclo corresponde en diseñar e integrar una prueba con otras herramientas (traceroute, iperf, etc) para conseguir puntos comunes y generar un enfoque de integración sencillo de los wrappers futuros a las aplicaciones. (ej. Lidiar con aplicaciones que requieren enfoque cliente solo [ping] o cliente-servidor [iperf]).

El cuarto ciclo consiste en hacer análisis del rendimiento del sistema y hacer las optimizaciones necesarias para ofrecer una calidad de servicio apropiada, determinar costos, limitaciones y requisitos mínimos para implementar en países en vías de desarrollo.

## 1.6 Alcance

## 1.7 Estructura del Documento

# Capítulo 2

# Marco Teórico

En este capitulo se presentan los conceptos necesarios para la comprensión de este documento y se describen las herramientas de software, métodos y formatos que se emplearán para el desarrollo del sistema propuesto.

## 2.1 Monitoreo de Redes

El monitoreo de redes se refiere al uso de sistemas computacionales para determinar el estado de redes de computadoras, el estado de la red puede ser visto como el conjunto de factores o métricas que describen la red en un momento dado, se hablará a detalle de algunas de estas métricas en la sección 2.3.

Mientras una red tiene ciertos elementos relativamente invariables como la posición de sus nodos, la longitud de los enlaces, el tipo de enlace (fibra óptica, cable, satelital, etc), el ancho de banda de los enlaces, la velocidad de procesamiento de los enrutadores, entre otros, el estado de la red viene determinado por elementos generalmente impredecibles, como condiciones climatológicas, problemas de configuración, equipos en mal estado y congestión, determinar todo este tipo de problemas para generar alertas, aplicar acciones correctivas o al menos tener conocimiento de ellas es el objetivo de un sistema de monitoreo de redes.

Existen dos paradigmas fundamentales en el monitoreo de redes, el monitoreo activo que consiste en generar tráfico y observar la respuesta de la red y el monitoreo pasivo,

que consiste en observar el tráfico que atraviesa un cierto enlace o nodo. Para tener una imagen completa de la red es conveniente usar ambos paradigmas ya que ambos pueden ofrecer distintas perspectivas.

## 2.2 Principio Fin-a-Fin

El principio de fin-a-fin (también llamado argumento de fin-a-fin) sugiere que las funciones en los niveles bajos de un sistema pueden ser redundantes o de poco valor comparadas con el costo de proveerlas a bajo nivel[7], en otras palabras este argumento recomienda que la complejidad de un sistema de computación distribuido debe estar en los nodos mas externos o de "mayor nivel".

Este principio es valioso para el diseño de cualquier tipo de sistema distribuido sin embargo ha sido fundamental en la expansión y desarrollo de redes de computadoras de propósito general y se sustenta en que los nodos intermedios de una red deben proveer solo las funciones necesarias para permitir la comunicación entre los nodos finales, de esta manera es posible implementar nuevas funcionalides en los extremos de la red sin necesidad de hacer cambios a los nodos intermedios.

En el contexto de monitoreo de redes, se denomina tomografía de redes al proceso de inferir aspectos internos de una red a partir de mediciones realizadas en sus nodos externos.

## 2.3 Métricas de calidad de un enlace

A continuación se explican las métricas mas comúnmente utilizadas para determinar la calidad de un enlace.

### 2.3.1 Latencia

La latencia es el tiempo que le toma a un paquete o mensaje viajar desde su origen hasta el punto destino [8]. Teóricamente la latencia está relacionada directamente con la distancia entre los puntos finales de una comunicación, en la practica los paquetes viajan a través de una red de enrutadores que retransmiten el mensaje hasta su destino,

la latencia total será la suma de cada uno de los siguientes factores para cada uno de los enrutadores que atraviese el paquete[8]:

- **Retraso de Propagación:** es el tiempo que tarda un mensaje en viajar del emisor al receptor y es función de la distancia por la velocidad a la que la señal se propaga
- **Retraso de Transmisión:** es el tiempo que tarda el emisor en poner todos los bits de un mensaje en el medio de transmisión y es función de la longitud del paquete y el ancho de banda del enlace
- **Retraso de Procesamiento:** tiempo requerido para revisar la cabecera del paquete, buscar errores y determinar el destino del paquete.
- **Retraso de Colas:** Cantidad de tiempo que un paquete pasa en la cola de una interfaz esperando su turno por ser procesado.

### 2.3.2 Tiempo de ida y vuelta

El tiempo de ida y vuelta (RTT por sus siglas es ingles) es el tiempo que le toma a un paquete viajar desde el origen a su destino y de vuelta, podría pensarse que el RTT es aproximadamente el resultado de multiplicar la latencia por dos, sin embargo existen factores como el retraso por procesamiento en el equipo destino o diferencias en el enrutamiento del paquete en su camino de vuelta que hace que sea arriesgado establecer esa proposición.

Uno de los métodos mas populares para medir el RTT es enviar un paquete ICMP Echo Request y esperar el correspondiente ICMP Echo Reply sin embargo también es posible medir el RTT pasivamente durante la transmisión de un flujo TCP usando marcas de tiempo en la cabecera de los mensajes TCP[9].

Es importante notar que no es lo mismo medir el RTT desde la capa de red con el protocolo ICMP que hacerlo en la capa de transporte con TCP o en la capa de aplicación con un protocolo como HTTP, evidentemente el RTT será mayor en las capas superiores, sin embargo estas métricas también son útiles ya que se acercan mas fielmente a la experiencia del usuario.

### 2.3.3 Perdida de paquetes

La pérdida de paquetes ocurre cuando los paquetes en una transmisión fallan en llegar a su destino, ya sea por interferencias en el medio de transmisión (por ejemplo obstáculos físicos en un enlace WiFi), o cuando son desechados por un nodo de la red, los paquetes pueden ser desechados si se determina que están corruptos o mas comúnmente debido a escenarios de congestión en los que un enrutador está recibiendo paquetes a una tasa mayor de la que puede retransmitirlos y no tiene mas opción que desechar los paquetes que desborden la cola.

La perdida de paquetes afecta dramáticamente la latencia percibida en la capa de aplicación, según [9] una perdida de paquetes del 5% puede introducir un retraso de medio segundo en la aplicación.

Mientras puede parecer que la perdida de paquetes es siempre producto de un problema, esta juega un papel vital en el algoritmo de prevención de congestión (congestion avoidance) del protocolo TCP, ayudándolo a detectar congestión en la red el cual responderá limitando su tasa de envío de datos, esto ha sido esencial para evitar el colapso de las redes ip NOTA: Nosotros alguna vez hablamos de esto, valdría la pena nombrar el RFC que lo explica.

### 2.3.4 Jitter

Se llama jitter a la fluctuación de la latencia durante la transmisión de un conjunto de paquetes, estas fluctuaciones vienen dadas principalmente por la variación del retraso que los paquetes experimentan en las colas en los enrutadores[1], sin embargo todos los factores mencionados en la sección 2.3.1 pueden contribuir en menor medida.

El término jitter puede tener distintas connotaciones sin embargo es usado frecuentemente por científicos en el área de la computación como la variación de una métrica (comúnmente latencia) con respecto a otra métrica de referencia (como latencia mínima o promedio), a esto también se le llama variación en el retraso de los paquetes (PDV por sus siglas en ingles), este termino es a veces preferido por ser mas preciso [10]

El jitter puede afectar considerablemente transmisiones en tiempo real como VoIP

o streaming sin embargo la correcta implementación de políticas de buffering del lado del receptor puede minimizar e incluso mitigar este efecto[1]

### 2.3.5 Throughput

En términos de redes de computadoras, el throughput es la tasa en bits por segundo a la que fluyen los datos a través de un enlace [1], el throughput puede compararse al caudal de un rio, donde mientras mas ancho sea el rio mas agua puede fluir a través de el.

El máximo throughput teórico entre un par de nodos esta determinado principalmente por el segmento de la red con menor ancho de banda, este comportamiento se ilustra en la figura 2.1, donde el cable entre el punto de acceso wifi y el ISP resulta ser el cuello de botella en la comunicación. En la practica el throughput también viene determinado por varios factores como otros flujos de datos con los que se comparte la red o la velocidad a la que el receptor es capaz de procesar el flujo de datos entrante.

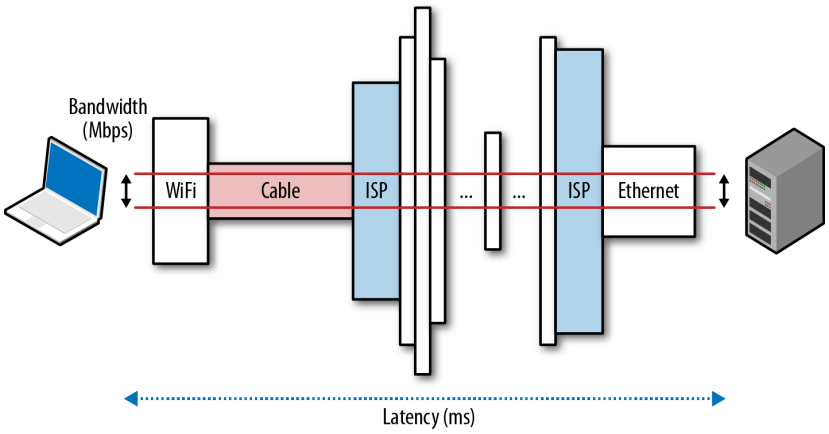


Figura 2.1: Latencia y ancho de banda – Imagen tomada del sitio: http://chimera.labs.oreilly.com/books/1230000000545/ch01.html

Para aplicaciones en tiempo real es esencial tener un throughput mínimo mayor a un cierto umbral que asegure que los datos se estén recibiendo a una tasa mayor o igual a la que se están consumiendo[1], es decir que a partir de esa tasa mínima un mayor throughput no implica una mejoría en la calidad de la comunicación, otras aplicaciones

como transferencias de archivos se benefician del mayor throughput posible ya que el tiempo de transferencia es una función del tamaño del archivo entre el throughput durante la transferencia.

No es posible medir el throughput de un enlace de forma pasiva, por lo que las pruebas que existen consisten en inyectar tráfico al enlace hasta saturarlo y determinar la velocidad a la que se reciben los datos del lado del receptor. Existen múltiples maneras de saturar el enlace, por ejemplo speedtest.net [11] utiliza hasta cuatro hilos paralelos transmitiendo mensajes aleatorios a través de HTTP, utilizar múltiples flujos es una forma efectiva de mitigar el efecto de la latencia sobre el throughput.

A diferencia de las métricas anteriores medir el throughput solo es posible con la colaboración de los nodos extremos del enlace, por lo que es necesario tener algún tipo de software preparado para aceptar la prueba e informar del resultado obtenido.

### 2.3.6 Uso del ancho de banda

Se le llama uso del ancho de banda a la medida de los datos que fluyen a través de un enlace, generalmente se busca optimizar el uso del ancho de banda para aprovechar al máximo los recursos de red.

Mas allá de solo determinar la cantidad de datos o la velocidad del flujo a través de un enlace, también es útil hacer análisis detallados del uso del ancho de banda para descubrir patrones de uso de la red, por ejemplo una universidad podría desear saber que porcentaje del ancho de banda esta siendo utilizado por streams de audio o video, a una empresa le gustaría saber cuanto tiempo pasan sus empleados en redes sociales, un analista desea saber cuales son los sitios web mas visitados en una red, etc.

NOTA PARA EL PROFESOR: ayuda, no encontré ninguna cita que me ayudara a definir estos conceptos, sin embargo me parece que es importante incluir esto en el marco teórico.

### 2.3.7 Disponibilidad

Un servicio esta disponible para un cliente cuando dicho cliente puede comunicarse con el, análogamente un servicio no esta disponible para un cliente cuando no puede

comunicarse con el, ya sea por un problema en el nodo final o en la red [12].

Generalmente la disponibilidad se mide a través de la disponibilidad promedio, que es la fracción del tiempo durante el cual un servicio esta disponible para un cliente promedio [12], sin embargo también se puede usar la disponibilidad continua, en este trabajo llamaremos actividad continua a un evento durante el cual un servicio está continuamente disponible para un cliente, y un inactividad continua a un evento durante el cual un servicio esta continuamente no disponible para un cliente.

Hemos definido la disponibilidad como un concepto meramente binario, en el que si un servicio es alcanzable entonces está disponible [12] sin embargo hay ciertos escenarios en los que se puede considerar que un nodo alcanzable está fallando en ofrecer un servicio adecuadamente, por ejemplo cuando un servidor web está respondiendo a las peticiones con un código de estado 500, o cuando la latencia es tal que hace que una comunicación en tiempo real no sea posible.

### 2.3.8 Bufferbloat

Bufferbloat es la existencia de buffers excesivamente grandes y generalmente llenos presentes en Internet [13]; puede parecer contra-intuitivo ya que buffers mas grandes implican que menos paquetes serán desechados al llegar a un enrutador congestionado, pero mientras la cola en la interfaz del enrutador crece también lo hace el tiempo de espera del paquete y a la vez interfiere (o invalida) los algoritmos de control de congestión de los protocolos mas comunes en la capa de transporte [13].

El bufferbloat puede ser mitigado configurando apropiadamente el hardware disponible, sin embargo es difícil de diagnosticar y es confundido frecuentemente con congestión en la red.

Recientemente se ha comenzado a medir el blufferbloat como el tiempo adicional que toma enviar paquetes a través de un enlace congestionado, algunas pruebas disponibles en linea [14][15] intentan determinar este retraso haciendo mediciones constantes de latencia al mismo tiempo que inundan el enlace para determinar la forma en que esta varía durante la prueba.

## 2.4 Big Data

Debido a la rápida evolución en la capacidad de almacenamiento y procesamiento de los sistemas computaciones y la adopción de los mismos por parte de miles de millones de usuarios en Internet, así como la creciente de popularidad de objetos inteligentes (Internet de las Cosas), sensores, cámaras, micrófonos, lectores biométricos, lectores de radiofrecuencia, entre muchos otros, día a día se están produciendo datos de forma rápida y masiva, esta tendencia creciente en la generación de datos ha hecho evidente la necesidad de tener sistemas capaces de manipular estos datos para obtener información que no se hubiera hecho evidente usando otros métodos de análisis.

Big Data se refiere la tendencia reciente de recolectar, almacenar y hacer análisis sobre cantidades masivas de datos, Big Data es un termino relativamente impreciso ya que no existe un convenio sobre la cantidad de datos a la que se refiere, pero usualmente el termino se relaciona con datos en el orden de los petabytes ($10^{15}$ bytes) y exabytes ($10^{18}$ bytes) [16].

El aprovechamiento de este flujo de datos generalmente inmanejable por los sistemas previamente concebidos ha sido adoptado por muchos tipos de organizaciones, por ejemplo las redes sociales rutinariamente analizan a sus usuarios para descubrir sus gustos y preferencias y ajustar cuidadosamente la publicidad que estos ven, juegos en linea analizan a sus jugadores para entender que factores determinan su comportamiento y de esta manera optimizar finamente las experiencias que les ofrecen, un ejemplo conocido de uso de big data en el ámbito medico ha sido la exitosa predicción de transmisión de enfermedades como el dengue a partir de patrones de búsqueda en google[17], también existe preocupación sobre el uso del big data para violar la privacidad de usuarios de Internet por parte de organizaciones gubernamentales de seguridad y vigilancia que son capaces de conocer todo tipo de detalles personales como transacciones y compras, sitios web visitados, búsquedas realizadas, publicaciones en redes sociales, posición geográfica con el uso de GPS, etc.

## 2.5 Visualización de datos

La visualización de datos se refiere al aprovechamiento de elementos gráficos para representar información cuantitativa; cualquier conjunto de datos no tienen significado sin alguna manera de organizar y presentar los descubrimientos relevantes que se encuentran potencialmente ocultos dentro de estos.

Los humanos podemos comprender los datos de mejor manera cuando son presentados a través de imágenes y elementos gráficos que leyendo números en tablas y columnas[18], una visualización apropiada de los datos permite de forma efectiva preguntar y responder las preguntas relevantes a una organización, por ejemplo en el marco de un sistema de monitoreo de redes preguntas como "¿donde están apareciendo los cuellos de botella?" "¿que factores afectan el rendimiento de un enlace?" o "¿cuales son los patrones de uso de los usuarios de la red?"

Hay una variedad de métodos apropiados para visualizar distintos conjuntos de datos, por ejemplo, los datos discretos se pueden observar a través de gráficas de barras, los gráficos de redes pueden comunicar la relación entre distintos entes, los mapas son efectivos para desplegar información geográfica, los mapas de calor permiten comparar el rendimiento de una variable a través del tiempo, etc, cada una de estas visualizaciones puede ser enriquecida a través del uso creativo de colores y formas para agregar nuevas dimensiones a los datos representados, por ejemplo, es posible combinar distintos conceptos para lograr visualizaciones aun mas poderosas como mapas de calor superpuestos en mapas de esta manera se pueden expresar datos de variable al mismo tiempo que se da una idea de su posición geográfica.

En la figura 2.2 se puede observar como se usan distintos métodos para representar datos sobre crímenes como robo de vehículos, homicidios y asaltos con armas letales en un área del distrito de Columbia, un mapa es usado para mostrar la localización en que se ocurrieron los crímenes, en esta representación se puede observar rápidamente que zonas son mas propensas a cada tipo de crimen y obtener una idea de su frecuencia, a la izquierda se puede ver una gráfica de barras mostrando la cantidad de ocurrencias coloreadas por tipo de crimen y separados por día de la semana, de esta manera es posible no solo observar que crímenes son mas frecuentes sino que días y semanas son las peores. También se ofrecen herramientas para filtrar los crímenes por fecha, tipo

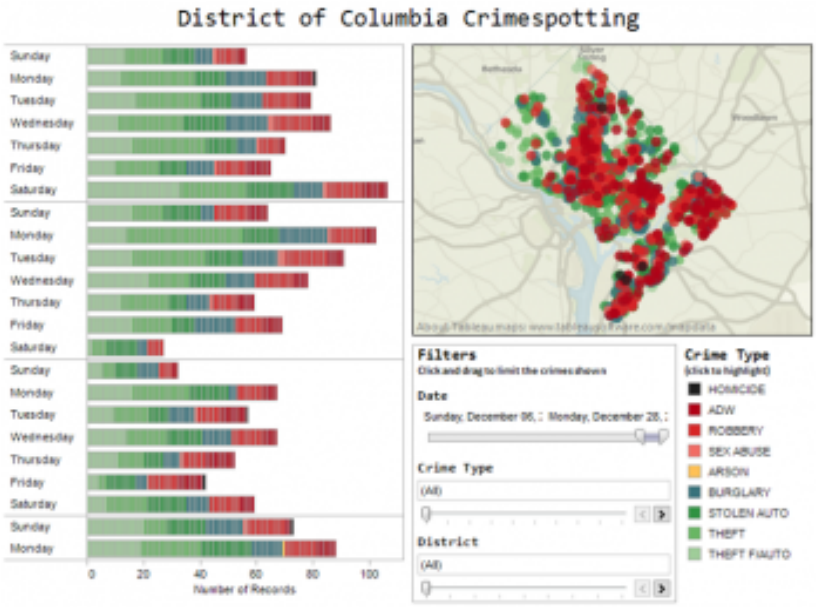


Figura 2.2: Visualizacion de datos sobre delincuencia en el distrito de Columbia – Imagen tomada del sitio:: http://www.pervasif.com/index.php/news-a-event/capabilities/data-visualization

de crimen y distrito, este tipo de herramientas de visualización de datos interactivas son cada vez mas populares en todo tipo de organizaciones ya que facilitan el análisis efectivo de sus datos.

## 2.6 Computación en la nube

La computación en la nube se refiere tanto a las aplicaciones desplegadas como servicios en Internet y el hardware y los sistemas computaciones en los centros de datos que proveen dichos servicios [19], a los servicios en si mismos se les ha dado el nombre de Software como Servicio (SaaS) y al hardware y software en los centros de datos es a lo que llamamos una nube. Cuando una nube se hace disponible para el publico en general a través de alguna forma de pago, se le llama una nube publica y el servicio que se esta vendiendo es Computación como Utilidad (Utility Computing), se le llama nube privada a los centros de datos internos de negocios u otras organizaciones pero que no pueden ser usados por el publico general, por lo tanto llamamos computación en

la nube a la suma de SaaS y Computación como Utilidad sin incluir nubes privadas[19]

La computación en la nube se caracteriza por ofrecer métodos de pago flexibles que permiten pagar solo por los recursos que se están utilizando (pay-as-you-go) y con una fina granularidad de modo que es lo mismo pagar mil procesadores una hora que un procesador por mil horas, esto permite a aplicaciones manejar cargas y escalas fluctuantes o patrones de uso específicos sin necesidad de tener hardware que esté ocioso durante largos periodos de tiempo. La nube ofrece a desarrolladores la ilusión de recursos computaciones ilimitados y disponibles a petición, eliminando la necesidad de planificar y aprovisionar equipos de hardware, y minimizar los riesgos relacionados con subestimar una aplicación que explota en popularidad o sobrestimar una aplicación que no llena las expectativas, en otras palabras no es necesario tener un gran capital de inversión inicial independientemente de la escala que resulte necesario manejar a corto o mediano plazo.

### 2.6.1 Modelos de servicio en la nube

Existen tres modelos de servicios en la nube que forman una "arquitectura orientada a servicios", estos son:

#### Infraestructura como Servicio (IaaS)

Infraestructura como Servicio a veces llamado Hardware como Servicio (HaaS) ofrece funciones básicas de almacenamiento y capacidad de computo como servicio ya sea a través de equipos de hardware físicos o de forma mas común como maquinas virtuales y una larga gama de imágenes de software disponible.

Esta capa abstrae al usuario de los detalles de infraestructura de los recursos de computo físico, localización, configuración, escala, seguridad, respaldo, mantenimiento, etc.

#### Plataforma como Servicio (PaaS)

Plataforma como servicio ofrece como servicio una plataforma para el desarrollo, ejecución y manejo de aplicaciones web, los recursos computacionales demandados por

la aplicación son manejados automáticamente, de modo que la complejidad de manejar la infraestructura subyacente queda eliminada, esto permite reducir enormemente la complejidad necesaria para desplegar aplicaciones, que pueden pasar de la etapa de desarrollo y pruebas rápidamente a un entorno de producción con un esfuerzo mínimo.

#### Software como Servicio (SaaS)

En el modelo de Software como servicio, los usuarios ganan acceso y hacen uso de aplicaciones de software, generalmente este tipo de software se paga bajo una subscripcion o por uso, este tipo de aplicaciones en la nube es conveniente y atractiva para los usuarios ya que estos no necesitan instalar software adicional en sus dispositivos sino que puede acceder a la aplicación desde navegadores web, esto permite que los usuarios puedan tener la misma experiencia en cualquier plataforma y reduce los costos operacionales.

## 2.7 Herramientas usadas para el desarrollo del sistema

A continuación se describen las herramientas usadas para el desarrollo del sistema de monitoreo de redes orientado a la recolección masiva de datos.

### 2.7.1 Modelo Vista Controlador (MVC)

El Modelo Vista Controlador es un patrón de diseño que divide la lógica de los datos y la presentación de forma claramente identificable y bien definida [20].

Este patrón de diseño es especialmente popular en el marco de aplicaciones web ya que su abstracción permite escribir software altamente desacoplado y fácil de mantener y escalar.

#### Modelo

Según [20] *"El modelo es un conjunto de clases que representan la información del mundo real que el sistema debe procesar, así por ejemplo un sistema de administración*

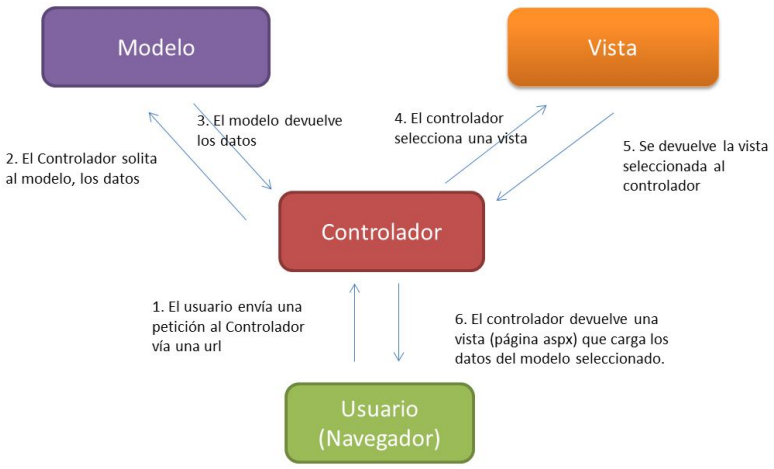


Figura 2.3: Ejemplo de la relación entre los componentes de una aplicación MVC – Imagen tomada del sitio: http://mind42.com

*de datos climatologías tendrá un modelo que representará la temperatura, humedad ambiental, estado del tiempo esperado etc"*

Según la implementación de MVC el modelo puede dividirse en el modelo del dominio que es el modelo propiamente dicho, es decir una colección de clases que modelan la realidad relevante a la aplicación, y opcionalmente el modelo de la aplicación; este modelo tiene conocimiento de las vistas y es capaz de enviar notificaciones cuando ocurren cambios en el modelo. El modelo de la aplicación también es llamado coordinador de la aplicación [20] .

### Vista

La vista es la encargada de determinar que información contenida en el modelo mostrar al usuario y la presentación, por ejemplo si se está modelando una caldera es posible tener una vista que dibuje gráficamente el nivel de la caldera y un termómetro con su temperatura y otra vista que sencillamente muestre estas propiedades en una tabla.

La vista pudiera cambiar cuando se actualiza el modelo del dominio a partir de notificaciones emitidas por el modelo de la aplicación, siguiendo con el ejemplo anterior de esta forma sería posible monitorear en tiempo real el estado de la caldera a partir

de sensores que mantengan actualizado el modelo.

#### Controlador

El controlador es el encargado de dirigir el flujo de control de la aplicación a partir de mensajes externos, como datos introducidos por el usuario en el caso de una aplicación de escritorio o peticiones HTTP en el caso de aplicaciones web. A partir de estos estímulos, el controlador se encargará de invocar las vistas apropiadas, actualizar el modelo y hacer todas las acciones necesarias.

Distintas implementaciones del patrón MVC se toman la libertad de establecer la linea que separa el controlador y la vista de forma distinta, por ejemplo en algunas implementaciones el controlador se encarga tanto de actualizar el modelo y las vistas como de responder a los mensajes externos, es decir que el controlador es el encargado de ejecutar toda la lógica, y la vista meramente contiene la presentación de los datos, en otras implementaciones como veremos en la próxima sección la vista es la encarga tanto de seleccionar los datos que van a mostrarse así como de desplegar la presentación, esta ultima es una variación de MVC a veces llamado MTV (Modelo-Template-Vista)

### 2.7.2 Django

Django es un framework de desarrollo de aplicaciones web construido en python, en este trabajo usamos Django como el fundamento para Octopus Head. Django permite construir aplicaciones web rápidamente gracias a su filosofía de "baterias incluidas", es decir, que incluye una inmensa gama de características comunes a la mayoría de las aplicaciones web como validaciones de formularios, autenticación de usuarios, manejo de sesiones entre muchos otros; de esta manera el desarrollador puede concentrarse en escribir la lógica que es especifica a su aplicación y dejar que Django maneje los aspectos repetitivos y muchas veces tediosos de la pila de desarrollo web.

Django esta diseñado con una arquitectura MVC es decir que separa claramente la lógica, de los datos y la forma en que dichos datos son presentados al usuario, en el caso de Django la "vista" describe que datos son presentados al usuario y el "template" representa la forma en que los datos son presentados.

Cuando Django recibe una petición esta pasa por un despachador de URLs (URL Dispatcher), cuya tarea es emparejar el URL con una vista y delegar a la vista el manejo de la petición. La vista contiene la lógica necesaria para atender la petición entrante, generalmente esto consiste en retirar o actualizar algunos datos del modelo, que a su vez se comunica con el manejador de base de datos, finalmente la vista combina los datos retirados de la base de datos, la petición y la sesión activa con una plantilla (template) para generar la respuesta que será devuelta, en la figura 2.4 puede verse el ciclo de petición-respuesta tal como se ha descrito.

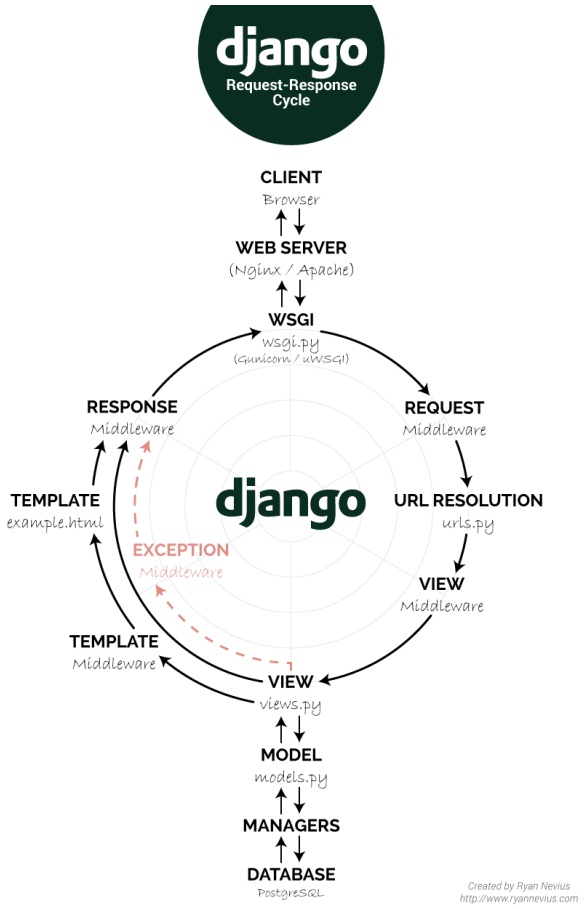


Figura 2.4: Ciclo de peticion-respuesta de Django

Las respuestas generadas por Django pueden ser tanto paginas HTML con CSS y Javascript pensadas para la interacción con el usuario así como respuestas en formato json o xml para la construcciones de APIs que pueden ser usados para la comunicación

maquina-maquina, esto es especialmente útil cuando se desea desarrollar aplicaciones en otras plataformas como Android o iOS que compartan el mismo backend.

Django ha demostrado ser escalable y flexible, se sabe de instancias de Django atendiendo ráfagas de cincuenta mil peticiones por segundo, ademas es de código abierto, gratuito y cuenta con una enorme comunidad de colaboradores y amplia documentación.

### 2.7.3 Celery

Celery es un sistema de procesamiento de tareas asíncrono, que permite tanto el encolamiento de tareas en tiempo real así como la planificación de tareas para ser ejecutadas mas tarde. Celery en realidad no implementa la mayoría de sus componentes, sino que define un protocolo de comunicación entre una serie de componentes (tambien llamados micro-servicios) que son:

- Bróker de mensajería: el message broker es el encargado almacenar las tareas que deben ser ejecutadas, este consiste de una o mas colas de prioridad, de esta manera es posible controlar finamente que tareas deben ser ejecutadas con mayor urgencia, y por que ejecutores.

- Planificador: el planificador es el encargado de encolar tareas previamente planificadas para que sean ejecutadas en algún momento especifico, el planificador no puede asegurar que las tareas sean ejecutadas exactamente en el tiempo planificado ya que el tiempo de ejecución depende del tamaño de la cola y de las tareas que ya estén siendo ejecutadas por los ejecutores.

- Workers (Ejecutores): los ejecutores son los encargados de ejecutar las tareas; estos se subscriben a una o mas colas y consumen los mensajes según la prioridad establecida en la cola finalmente escriben los resultados obtenidos en un result backend

- Result Backend: sirve como un almacen donde se guardan los resultados de las tareas ejecutadas durante un periodo de tiempo dado, esto hace posible consultar los resultados obtenidos por un tarea

Uno de los problemas comunes en el desarrollo de aplicaciones web es la ejecución de tareas largas o altamente bloqueantes que mantienen ocupado al servidor web durante periodos prolongados y peor aun mantienen al usuario final esperando, para evitar este escenario el servidor web debe tener una forma de delegar este tipo de tareas para ser ejecutadas después y responder inmediatamente.

Celery es usado comúnmente junto a Django y permite a una aplicación web escalar indefinidamente ya que solo hace falta aumentar el número de trabajadores disponibles que se pueden distribuir en tantas maquinas físicas (o maquinas virtuales en un servicio IaaS) como sea necesario.

### 2.7.4 Redis

Redis es un almacén de estructuras de datos en memoria que puede ser usado como base de datos, cache o bróker de mensajería; se caracteriza por su velocidad de respuesta ya que todos sus datos se mantienen en memoria principal y no invierte recursos en asegurar la persistencia de sus datos, evidentemente esto tiene como consecuencia que se pierden sus datos almacenados en caso de que ocurra cualquier falla inesperada. Redis puede ser usado como bróker de mensajería o como result backend por una aplicación Celery.

### 2.7.5 Bases de Datos

Uno de los pilares fundamentales de casi toda aplicación moderna es tener un modo de almacenar datos de forma persistente en el tiempo así como consultarlos y actualizarlos de forma rápida, segura y resistente a fallas.

Según [21] una base de datos es una colección de datos estructurados. Puede ser cualquier cosa desde una simple lista de compras, una galería de fotos o las bastas cantidades de información en una red corporativa. Para agregar, acceder y procesar la data almacenada en una base de datos, se necesita un sistema manejador de base de datos. Ya que los computadores hacen un muy buen trabajo manejando grandes cantidades de datos, los sistemas manejadores de bases de datos juegan un papel central en la computación como utilidades independientes o partes de otras aplicaciones.

SQL por sus siglas en ingles "Structured Query Language" (lenguaje de consulta estructurado) es el lenguaje estandarizado mas común para acceder a bases de datos, SQL está definido por el Estándar ANSI/ISO SQL y ha ido evolucionando desde 1986 para convertirse en un estándar de facto en el mundo de la computación.

### Mysql

Mysql es un sistema manejador de bases de datos relacionales (RDBMS por sus siglas en ingles) de codigo abierto bajo la licencia GPL (GNU General Public License). Mysql se caracteriza por ser rápido, confiable, escalable y fácil de usar, es posible instalar Mysql tanto en una maquina junto a otras aplicaciones como servidores web o también instarlo en maquinas dedicadas para que use todo el poder disponible. Mysql posee características para ejecutarse en clusters de maquinas junto con un motor de replicacion para obtener una alta escalabilidad [21].

### Mapeo Objeto-Relacional

El Mapeo Objeto-Relacional (ORM por sus siglas en ingles) es un método para interactuar con bases de datos relaciones desde el paradigma de la programación orientada a objetos, de esta manera es posible aprovechar conceptos como herencia y polimorfismo.

Según [22] la mayoría de las aplicaciones modernas usan lenguajes orientados a objetos como Java o C# para construir aplicaciones y bases de datos estructuradas para almacenar datos, por lo tanto, es util tener una interfaz que transforme los datos entre estos tipos de incompatibles.

El uso de un ORM simplifica enormemente el manejo de la estructura de datos subyacente ya que permite al programador manejar los datos a un mayor nivel de abstracción como si fueran objetos, sin necesidad de generar manualmente las consultas SQL, ademas esta capa de abstracción permite desacoplar el código de la aplicación de los detalles específicos de cada RDBMS.

### 2.7.6 Json (JavaScript Object Notation)

Json (JavaScript Object Notation) o notación de objectos javascript es un formato textual de intercambio y almacenamiento de datos no estructurados, json posee un formato que es fácil de leer para humanos y fácil de interpretar para maquinas y mas ligero que XML por lo que se ha popularizado para el desarrollo de APIs.

Json soporta dos tipos de estructuras de datos:

1. Colecciones de pares nombre,valor comparable a un diccionario o tablashash.
2. Listas ordenadas de valores, similar a las listas o vectores que existen en virtualmente cualquier lenguaje de programación

Un archivo en formato json puede estar formado de cualquiera de las estructuras de datos antes escritas o cualquier permutacion de dichas estructuras anidadas.

Los tipos de datos soportados por json son:

- Number: numeros flotantes de doble presicion en formato Javascript
- String: cadenas de caracteres unicode encerradas en dobles comillas
- Boolean: true o false
- Arreglo: secuencia de valores ordenados
- Objeto: colección no ordenada de pares nombre,valor
- null: nulo o vacío

### 2.7.7 Dropbox

Dropbox es una plataforma de almacenamiento de datos en la nube de tipo PaaS y SaaS que permite compartir y sincronizar archivos entre un número arbitrario de clientes.

Dropbox es usado por aproximadamente 400 millones de personas y 100000 organizaciones[23] y posee aplicaciones en Windows, Linux, Mac OS X, iOS, Android, Blackberry y web.

Como todo servicio en la nube es atractivo para desarrolladores por ser robusto y confiable y es gratis hasta cierta cuota de almacenamiento a partir del cual el pago escala según la cantidad de almacenamiento usado.

#### API de Dropbox

Un API (Aplication Programming Interface) o interfaz de programación de aplicaciones, es una serie de métodos o funciones orientados a la comunicación maquina-maquina, en el caso de la computación en la nube un API conforma un servicio que permite desarrollar aplicaciones una plataforma (en este caso Dropbox).

El API de Dropbox permite realizar peticiones (como subir o descargar archivos, listar directorios o crear carpetas) sobre el espacio de almacenamiento de un usuario, el usuario debe previamente dar permiso a la aplicación para que esta pueda hacer cambios a su nombre, el usuario puede elegir denegar el acceso a la aplicación en todo momento y existen distintos tipos de esquemas de acceso donde una aplicación solo tiene acceso a un conjunto limitado de directorios dentro del espacio de almacenamiento del usuario.

El API de Dropbox impone ciertos límites de peticiones por usuario para impedir que una aplicación realice una cantidad excesiva de peticiones durante un cierto periodo de tiempo, sin embargo el limite se considera lo suficientemente alto como para no entorpecer la inmensa mayoría de los casos de uso.

### 2.7.8 Diseño web adaptable

Debido a la inmensa diversidad de dispositivos desplegados en el mercado y sus distintos tamaños y formas es imposible realizar manualmente diseños que puedan ajustarse a cada uno de ellos, anteriormente una solución popular a este problema era tener varias versiones con distintas resoluciones y elegir que versión mostrar a cada cliente, sin embargo esto ya no es necesario gracias a las nuevas herramientas disponibles en HTML5 CSS y Javascript que están ampliamente implementadas en navegadores modernos.

El diseño web adaptable o "Responsive Web Design" es la tendencia en el diseño de paginas web que se ajusten elásticamente a cualquier resolución, adaptando la forma

en que se presentan sus elementos de forma "inteligente".

Las técnicas mas comunes para lograr esto es tener elementos que ocupen el mayor espacio horizontal posible en pantallas grandes y mientras el tamaño horizontal se reduce estos pasan a ocupar el espacio verticalmente; siempre ocupando el máximo del ancho disponible, evitando crear barras de desplazamiento horizontal que desorientan e incomodan a los usuarios.

Otra heuristica en la creación de sitios web adaptables es escalar o esconder elementos gráficos decorativos o menús de navegación laterales, reducir el tamaño de margenes o incluso cambiar tipos de letras para que sean mas legibles en dispositivos móviles, mientras el tamaño del dispositivo es menor, cada pixel se vuelve mas precioso.

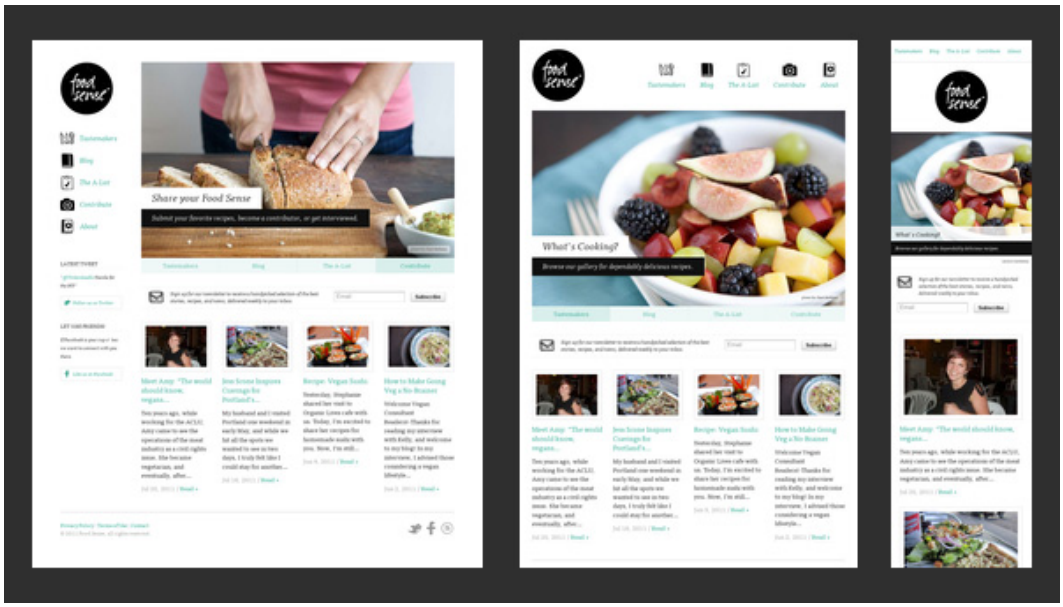


Figura 2.5: Ejemplo de diseño adaptable, se puede ver como los elementos se reagrupan, escalan o esconden para ajustarse al tamaño de la pantalla.

El diseño web adaptable no solo facilita el desarrollo de sitios web ahorrando a los desarrolladores y diseñadores el costo de construir múltiples versiones de un mismo sitio web sino que ademas el sitio web ofrece una experiencia similar independientemente del dispositivo con el que se este visitando.

### 2.7.9 Highcharts

Highcharts es una biblioteca Javascript para dibujar gráficas en entornos HTML es gratis para proyectos no comerciales y de código abierto. Las gráficas generadas por highcharts aprovechan las características de HTML5 por lo que pueden soportar enormes conjuntos de datos sin afectar el rendimiento incluso en dispositivos móviles, son totalmente interactivas de manera que el usuario puede inspeccionar detalladamente el conjunto de datos y son dinámicas permitiendo actualizar la gráfica en tiempo real cuando se reciben nuevos datos.

Highcharts incluye una amplia gama de gráficas predefinidas que la hace ideal para todo tipo de visualización de datos como mapas de calor, splines, gráficas de área, barras, pie, mapas, entre muchos otros; cada una de ellas ofrece un gran control sobre la forma en que son presentadas de modo que es posible lograr casi cualquier resultado deseado.

Desde el punto de vista del programador es muy fácil de usar ya que solo requiere especificar un "objeto de configuración" y uno o mas arreglos conteniendo los datos a desplegar, es posible obtener distintas representaciones de los mismos datos solo cambiando el objeto de configuración.

### 2.7.10 Google Maps

Google Maps es el servicio de mapas web de Google, ofrece distintos tipos de mapas, como mapas viales, mapas de relieve e incluso imágenes satelitales e imágenes de calles en tercera dimensión (llamado Google Street View)

Los mapas de Google Maps son totalmente dinámicos, permitiendo al usuario desplazarse, hacer zoom y cambiar el tipo de mapa a voluntad, Google Maps funciona dividendo el espacio mostrado en sectores que son descargados individualmente, de manera que cuando el usuario desplaza el mapa solo es necesario descargar los nuevos sectores desde los servidores de Google.

Google Maps es una de las herramientas mas populares para dibujar mapas web ya que ofrece un API gratuito y fácil de usar que permite enmarcar mapas en cualquier sitio web con solo algunas lineas, ademas los mapas de Google son de altisima calidad

y se mantienen constantemente actualizados.

### 2.7.11 Geo-localizacion IP

La geo-localizacion IP consiste en asignar a un IP la localizacion geográfica de la maquina anfitriona correspondiente[24].

Existen dos paradigmas principales para aproximar la localización geográfica de una dirección IP: activo y pasivo; las técnicas activas de localización se basan en mediciones de retraso y en muchos casos proveen resultados precisos[24], el paradigma pasivo consiste del uso de bases de datos que contienen rangos de direcciones ips a los que se les llaman bloques o prefijos relacionados a una localización geográfica especifica, sin embargo su precisión puede estar sujeta a errores substanciales. En ambos casos es imposible conocer la localización exacta asociada a una dirección IP sin la colaboración activa de los anfitriones finales, sin embargo es posible hacer buenas aproximaciones en algunos a nivel de ciudades o países.

Ya que conocer la localización de sus clientes a partir de su dirección ip es útil para muchos servicios (por ejemplo, para conducir anuncios localizados) existe una gran variedad de soluciones de geolocalizacion tanto gratuitas como de pago.

### 2.7.12 Ajax

AJAX (Asynchronous JavaScript And XML) es una técnica para construir sitios web interactivos a través de peticiones asíncronas con Javascript que mantienen comunicación con el servidor web para mantener el estado del cliente actualizado sin necesidad de que el usuario tenga que refrescar la pagina o realizar consultas adicionales, de esta manera el usuario tiene una experiencia similar a la que tendría con aplicaciones de escritorio.

A pesar de que el nombre AJAX sugiera el uso XML como lenguaje para la transferencia de datos entre el cliente y el servidor, se puede usar cualquier formato como texto plano, HTML y JSON

### 2.7.13 APScheduler

APScheduler (Advanced Python Scheduler) es una biblioteca python para retrasar la ejecución de rutinas a un instante dado en el futuro ya sea como eventos de una sola vez o de forma recurrente[25], esta biblioteca provee las herramientas para construir cualquier esquema de planificación que se desee su arquitectura consta de componentes altamente desacoplados que pueden ser mezclados, combinados o incluso extendidos para ofrecer nuevas funcionalides.

#### Triggers (gatillos)

Los triggers (gatillos) son los encargados de determinar el momento de la próxima ejecución de una tarea, APScheduler incluye tres gatillos predefinidos: DateTrigger que ejecuta tareas dada una fecha y hora, IntervalTrigger que ejecuta tareas de forma recurrente dado un intervalo fijo y CronTrigger que ejecuta tareas de forma similar a crontab del sistema UNIX.

#### Job Stores (almacenes de tareas)

Job Stores o almacenes de tareas determinan la forma en que las tareas serán alojadas por el planificador, por defecto las tareas se guardan en memoria, sin embargo también es posible guardar las tareas en almacenes persistentes como bases de datos SQL o mongo.

#### Executors (ejecutores)

Los ejecutores son los encargados de ejecutar tareas y posteriormente informar al planificador del estado de la tarea. El ejecutor por defecto consta de un arreglo de hilos (thread pool) pre-instanciados listos para aceptar tareas, sin embargo si se tienen tareas de uso intensivo de CPU, esta disponible un ejecutor basado en procesos que puede aprovechar mejor las características de procesadores multinúcleos [25].

#### Shedulers (Planificadores)

Los planificadores son los que unen todos los elementos mencionados anteriores y son los encargados de gestionar las tareas, es decir que delegan a los ejecutores la ejecución de las tareas en el tiempo adecuado y duerme esperando la ocurrencia de eventos que requieran su atención.

APScheduler incluye dos planificadores predefinidos: BackgroundScheduler que se ejecuta instanciando un hilo en segundo plano y permite que el flujo del programa continúe, útil cuando se desea realizar otras funciones o como agregar o modificar tareas después de que el planificador se ha inicializado y BlockingScheduler que se ejecuta en el mismo hilo de ejecución bloqueando el flujo del programa indefinidamente.

### 2.7.14 Ping

Ping es un programa utilitario incluido en todos los sistemas basados en UNIX y Windows que comprueba la presencia y tiempo de respuesta de un host en una red IP. Ping utiliza el Protocolo de Mensajes de Control de Internet (ICMP) para enviar un paquete de solicitud ICMP (ICMP Echo Request) y espera el mensaje de respuesta del host remoto (ICMP Echo Reply); calculando la diferencia de tiempo entre el envío y la recepción se puede calcular la latencia de la red, ping también incluye funciones para enviar paquetes en ráfaga útil cuando se desea medir la perdida de paquetes.

Ya que históricamente ping se ha usado por atacantes para determinar la presencia de equipos en una red o realizar ataques de denegación de servicio (ping flood) muchos enrutadores y firewalls bloquean estos mensajes como medida de seguridad, aunado a esto ya que ping utiliza ICMP que es un protocolo de capa de red, este no es capaz de alcanzar equipos detrás de un NAT; a pesar de esto, ping ha demostrado ser una herramienta vital en el diagnostico y monitoreo de redes IP.

### 2.7.15 Traceroute

Traceroute (tambien llamado Tracert en sistemas Windows) es un programa utilitario de diagnostico que permite conocer los hosts que visita un paquete durante su transito por una red.

Al igual que Ping, Traceroute utiliza el protocolo ICMP pero envía paquetes con un valor de "Time to Live" (TTL) incremental, cada vez que un nodo de la red recibe un paquete decrementa su valor de TTL y si este llega a cero lo descarta y envía de vuelta al host emisor un mensaje de control indicando que el TTL llegó a 0, de esta manera Traceroute puede generar una lista de los nodos visitados y el valor de RTT para cada uno de ellos.

Un análisis cuidadoso de la salida de Traceroute puede ayudar a diagnosticar numerosos problemas en una red como ineficiencias en el enrutamiento, presencia de enrutadores congestionados, cuellos de botella, comportamientos inesperados, etc.

### 2.7.16 Iperf

Iperf es una herramienta que permite medir el rendimiento (throughput) entre un par de nodos en un red. Al igual que muchas otras pruebas para medir velocidad de transferencia, Iperf funciona con una arquitectura cliente-servidor, donde el cliente genera un flujo de datos hacia el servidor y mide la velocidad obtenida, también es posible ejecutar una prueba "en reversa" donde es el servidor el que genera el flujo de datos.

Iperf puede ejecutar pruebas utilizando TCP o UDP, sin embargo existen diferencias entre ellos:

- Ya que UDP a diferencia de TCP no implementa ningún algoritmo de control de congestión se podría obtener un rendimiento ligeramente superior con este protocolo *Nota para el profesor: yo se que esto es "verdad" pero me gustaría poder sustentarlo un poco mejor o incluir una cita
- Con UDP se puede obtener una estadística de la perdida de paquetes durante la prueba
- Con UDP es el servidor el que totaliza los resultados, ya que el cliente no tiene manera de saber que datagramas se han recibido.

Para obtener una buena medición del rendimiento del enlace hay que ajustar los parámetros de la prueba cuidadosamente, por ejemplo, es posible ajustar la cantidad

de datos que se van a transferir durante la prueba, elegir una cantidad muy pequeña podría resultar en que no sea suficiente para saturar el enlace, mientras tanto elegir una cantidad demasiado grande podría resultar en una prueba innecesariamente larga, en ambos casos el valor del rendimiento obtenido no reflejará la realidad, también hay que tomar en cuenta que es difícil obtener una lectura exacta del rendimiento del enlace ya que podrían existir otros flujos en la red que afecten el resultado de la prueba.

# Capítulo 3

# Monitor de Red "Tentacle Monitor"

Tentacle Monitor es un monitor de red ligero, diseñado para ajustarse a las limitaciones de sistemas de bajo costo y con la intensión de ser desplegado en las redes que se desea monitorear y ser dejado desatendido durante periodos arbitrarios de tiempo.

Tentacle Monitor ejecuta pruebas periódicas siguiendo un plan de monitoreo definido por el usuario con el objetivo recoger datos sobre las métricas relevantes a la red a monitorear, los resultados de las pruebas son guardados temporalmente en la memoria del dispositivo y se suben a la nube regularmente, la frecuencia en que se suben los datos a la nube depende de la cantidad de memoria disponible en el dispositivo, la cantidad de datos generados por el monitoreo y el ancho de banda disponible.

Uno de los objetivos principales de Tentacle Monitor es ejecutar las pruebas en el momento preciso dado por el plan de monitoreo, ya que el posterior análisis de los datos por parte de Octopus Head exige que las muestras se tomen con un patrón regular y conocido, para esto, un planificador que asegura la ejecución precisa de las pruebas es central en la arquitectura del monitor. Como ya se mencionó, el comportamiento del monitor viene dado por un plan de monitoreo, este es definido por el usuario en Octopus Head, toda la comunicación entre Tentacle Monitor y Octopus Head se realiza a través de la nube, Octopus Head transfiere el plan de monitoreo a través de archivos a modo de mensajes y los guarda en una carpeta a modo de buzón; Tentacle Monitor

mantiene una conexión con la nube para ser notificado de cambios en dicha carpeta con baja latencia de esta manera se pueden leer los mensajes rápidamente y tomar las acciones necesarias, como incluir nuevas pruebas, replanificar pruebas, cambiar los parámetros de ejecución, o incluir nuevos enlaces a monitorear.

## 3.1 Diseño del Monitor

El diseño del monitor esta sujeto a los siguientes baremos:

- **Estabilidad** ya que el monitor podría dejarse desatendido es esencial que sea estable, si el monitor no se está ejecutando el usuario no obtendrá los resultados esperados en la aplicación web.
- **Flexibilidad** el monitor debe ser capaz de cargar nuevas pruebas en tiempo de ejecución, esto es especialmente importante ya que no desea interrumpir otras pruebas que se estén ejecutando.
- **Planificación precisa** es importante que las pruebas se ejecuten en el momento adecuado, siguiendo de manera fiel el plan de monitoreo.
- **Ejecutable en equipos de bajo costo** el monitor debe incluir el código mínimo para su funcionamiento, tener un uso eficiente de memoria y evitar desbordar la memoria secundaria del host.
- **Configuración remota** el monitor debe incluir algún protocolo para actualizar su plan de monitoreo a partir de cambios hechos en la aplicación web.
- **Bitácora** ya que el monitor se ejecuta como un proceso daemon, se desea tener una bitácora donde se puedan leer mensajes sobre los eventos relevantes durante la ejecución.

### 3.1.1 Arquitectura

Como se puede observar en la figura 3.1, Tentacle Monitor tiene una arquitectura basada en componentes altamente desacoplados con responsabilidades bien delimitadas

para facilitar el desarrollo de la aplicación; cada uno de estos componentes puede ser reemplazado fácilmente siempre y cuando la interfaz entre ellos se mantenga intacta.

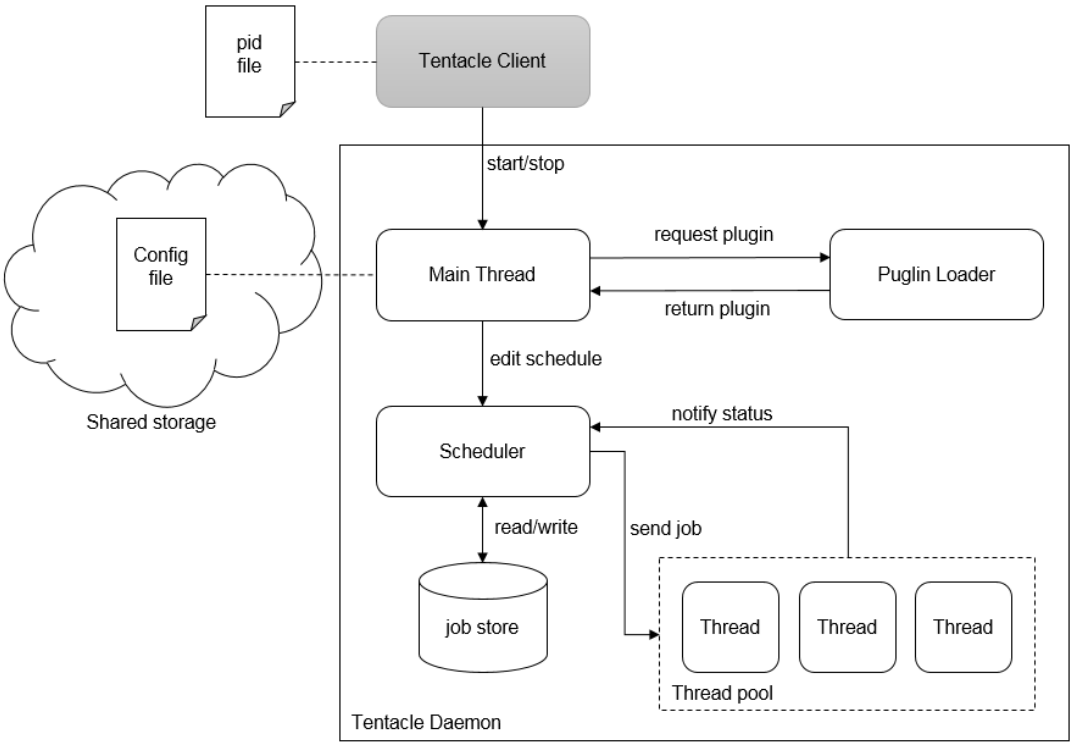


Figura 3.1: Diagrama de Componentes Octopus Tentacle

### 3.1.2 Diagrama de actividades

La ejecución del monitor comienza en el cliente, el cliente es el encargado de instanciar el proceso daemon y lo hace a través del método bien conocido de doble fork. Para asegurarse de no ejecutar múltiples instancias del monitor se revisa un archivo .pid, si el archivo existe significa que el monitor se está ejecutando y el cliente informa del error al usuario.

Después de que el proceso esta daemonizado comienza la fase de inicializacion, para esto se crea el planificador y se cargan las tareas del jobstore, importando el codigo de las pruebas a ejecutar. A este punto las pruebas están planificadas tentativamente y cargadas en memoria pero solo serán ejecutadas después de que se inicie el planificador.

Al iniciar el planificador este pasa a ejecutarse como un subproceso y se encarga de iniciar las tareas en el momento preciso. Cuando el planificador inicia una tarea se la pasa al ejecutor en este caso un grupo de subprocesos previamente inicializados, cuando un subproceso termina de ejecutar una tarea informa al planificador, esto con la finalidad de implementar políticas de concurrencia de tareas.

Mientras tanto, el hilo principal se conecta a la nube usando el API de Dropbox que posee un método para notificar a los clientes sobre cambios a una carpeta en tiempo real y con baja latencia, el método consiste en abrir una conexión HTTP con un alto valor de timeout (entre 30 y 120 segundos), si ocurre un cambio en la carpeta, el servidor de Dropbox responde de inmediato indicando que ocurrieron cambios y el monitor procederá a manejar este evento (descargando los archivos nuevos y aplicando los cambios al plan de monitoreo), en caso contrario, Dropbox responde indicando que no ocurrieron cambios y el monitor reinicia la conexión.

Los eventos relevantes al monitor tentacle son los siguientes: (1) una prueba se ha agrega al plan de monitoreo y debe cargarse a memoria y planificarse (2) una prueba se ha eliminado y debe ser eliminada del plan de monitoreo (3) el intervalo entre pruebas de una prueba ha cambiado y debe replanificarse la próxima ejecución (4) los parámetros de una prueba han cambiado (5) se ha agregado un nuevo enlace a monitorear (6) se ha eliminado un enlace monitoreado.

El monitor solo puede detenerse a través de una señal del sistema operativo (SIGTERM), el cliente utiliza el archivo pid para determinar el id del proceso y enviar la señal. El manejador de excepciones del monitor entonces inicia una secuencia de apagado, deteniendo las pruebas, apagando el planificador y desbloqueado el archivo .pid.

Esta secuencia de actividades puede verse en la figura 3.2.

## 3.2 Componentes

### 3.2.1 Cliente

El cliente de Octopus Tentacle es el programa encargado de iniciar o detener la ejecución del monitor en modo daemon y funge como interfaz entre el usuario y monitor.

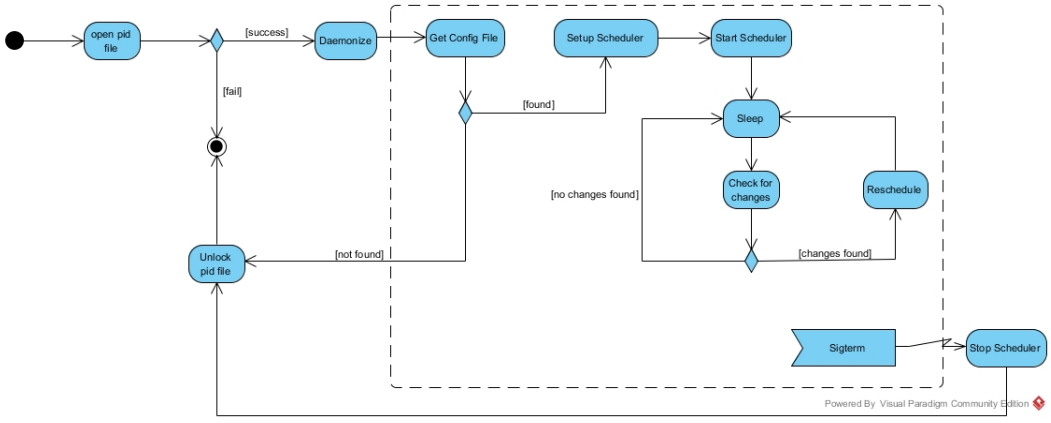


Figura 3.2: Diagram de actividades de Tentacle

Se puede invocar el cliente con los siguientes comandos:

| Comando | Descripción |
|---|---|
| start | Inicia el monitor como un proceso daemon, falla si el archivo .pid ya existe (el monitor se esta ejecutando) |
| stop | Detiene el monitor enviando la señal SIGTERM al proceso daemon, falla si el archivo pid no existe (el monitor no se esta ejecutando) |
| restart | Detiene e inicia el monitor |
| jobs | Muestra las tareas pendientes en el planificador (plan de monitoreo) |

Tabla 3.1: Comandos del cliente Octopus Tentacle

Los comandos start y restart se pueden invocar con la opción –no-daemon para iniciar el monitor en modo consola (sin daemonizar)

### 3.2.2 Hilo Principal

El hilo principal de ejecución es el punto de partida desde el momento en que el proceso ya se ha convertido en un daemon, se encarga de inicializar el planificador el cual a su vez pasa a ejecutarse en segundo plano, luego, su tarea consiste en mantener el plan de monitoreo al día a partir de los cambios que se hagan en Octopus Head.

El hilo principal mantiene una conexión constante con el servidor de Dropbox para

ser notificado de cambios en el plan de monitoreo, si hay cambios, este llama los métodos para leer los archivos y a partir de ellos hacer cambios en plan de monitoreo, como eliminar pruebas o agregar nuevos enlaces monitoreados.

### 3.2.3 Planificador

Para la implementación del planificador se hizo uso de APScheduler, una biblioteca que permite retrasar la ejecución de código python; el planificador se puede ejecutar como un subproceso de modo que es posible agregar, eliminar y re-planificar tareas en cualquier momento desde el hilo principal de la aplicación.

APScheduler consiste de un conjunto de componentes configurables que se pueden extender o re-usar para obtener cualquier comportamiento deseado, a continuación se explica su funcionamiento y como se usaron en el marco de esta aplicación.

#### Triggers (Gatillos)

Los triggers contienen la lógica para determinar en que momento se debe ejecutar una tarea, la biblioteca incluye varios triggers predefinidos, de los cuales dos se han usado para el desarrollo de esta aplicación:

- Interval Trigger: ejecuta pruebas en intervalos regulares, opcionalmente se pueden suministrar fechas finales e iniciales de modo que las pruebas solo se ejecuten durante un periodo especifico
- Date Trigger: ejecuta la prueba en una fecha especifica dada, una sola vez, útil para desplegar pruebas que usen muchos recursos de red como benchmarks de ancho de banda o capturas de tráfico de la red.

#### Executors (Ejecutores)

El ejecutor es el ente encargado de llevar a cabo la ejecución de las tareas, en nuestro caso se ha hecho uso de un grupo de subprocesos (thread pool), la cantidad de hilos en el grupo esta dado por el numero de pruebas que estaremos ejecutando de modo que siempre se tenga al menos un hilo disponible cuando se inicia una prueba.

#### Almacén de tareas (Job store)

El almacén de tareas guarda las tareas planificadas, el comportamiento por defecto es guardar las tareas en memoria, pero existen distintos tipos de almacenes como redis (ver sección 2.7.4) y bases de datos; hemos usado el SQLAlchemy job store que permite guardar tareas en una base de datos ligera como sqlite y así asegurar la persistencia de los datos de la aplicación.

### 3.2.4 Almacenamiento Compartido

El almacenamiento compartido se encarga de alojar los resultados de las pruebas en archivos de trazas, por cada prueba que se esté ejecutando se crea una carpeta donde se alojan sus respectivos archivos.

Para mantener el numero de archivos en el almacenamiento compartido razonablemente pequeño se crea para cada enlace un archivo por hora y todas las trazas que se generen en ese periodo se anexan al archivo correspondiente; el costo de alojar archivos en la nube no depende del numero de archivos sino del espacio total de disco en uso, ya que existe un limite de peticiones diarias por usuario debemos intentar minimizar la cantidad de peticiones que realizamos a Dropbox. Este tema se explicará a profundidad en la sección **??**.

## 3.3 Pruebas Implementadas

Gracias a la arquitectura de Tentacle, implementar una prueba es tan sencillo como crear un paquete e implementar la función "run" en el archivo init.py, todas las pruebas implementadas hasta ahora comparten una flujo de ejecución similar:

1. Se lee el archivo de configuración
2. Se obtienen los enlaces a monitorear y los parámetros globales
3. Se ejecuta la prueba por cada enlace monitoreado (ya sea en paralelo o en secuencia).

4. Se ejecuta un comando externo externo como ping o traceroute o se usa una biblioteca python para evaluar alguna métrica del enlace.

5. (opcional) si se ejecuta un comando externo se hace un parsing para extraer los resultados relevantes de la salida del programa

6. Se guardan los resultados en un archivo de trazas; generalmente el resultado de una prueba está representado por una linea en archivo de trazas, sin embargo el desarrollador tiene libertad total sobre el formato que utilice para generar archivos de trazas

Durante el desarrollo de este proyecto se han implementado las siguientes pruebas:

### 3.3.1 Ping

Esta prueba hace uso del comando ping para obtener datos de la latencia en un enlace, como ya se menciono en la sección 2.7.14 ping viene incluido en todas las distribuciones de linux por lo que no es necesario instalar ninguna dependencia o programa externo.

La prueba consiste en ejecutar el comando ping para cada uno de los enlaces monitoreados, ejecutamos el comando con la opción -D para que ping imprima cada resultado de latencia con una marca de tiempo entre corchetes, un ejemplo de la salida de ping se puede ver en la figura 3.3.

Es muy sencillo extraer los datos relevantes de la salida de ping, para esto recorremos la salida descartando las lineas que no comiencen con el carácter '[', separamos la salida en palabras, la palabra en la posición 0 corresponde a la marca de tiempo, luego buscamos las palabras que comiencen por "icmp_seq o icmp_req y time" para obtener numero de secuencia icmp y tiempo de ida y vuelta respectivamente.

Independientemente del número de sondas que se envíen elegiremos solo un resultado de latencia por prueba (la mediana), si para una prueba no se obtiene ninguna respuesta entonces guardamos una traza con rtt=-1, indicando que el enlace está inactivo o el nodo está rechazando el protocolo.

```
jesus@jesus-pc:~$ ping 150.185.138.59 -c 10 -i 1 -D
PING 150.185.138.59 (150.185.138.59) 56(84) bytes of data.
[1443758089.876135] 64 bytes from 150.185.138.59: icmp_seq=1 ttl=47 time=8485 ms
[1443758090.547867] 64 bytes from 150.185.138.59: icmp_seq=2 ttl=47 time=8148 ms
[1443758091.448875] 64 bytes from 150.185.138.59: icmp_seq=3 ttl=47 time=8041 ms
[1443758092.293731] 64 bytes from 150.185.138.59: icmp_seq=4 ttl=47 time=7878 ms
[1443758093.116296] 64 bytes from 150.185.138.59: icmp_seq=5 ttl=47 time=7693 ms
[1443758093.928141] 64 bytes from 150.185.138.59: icmp_seq=6 ttl=47 time=7497 ms
[1443758094.578914] 64 bytes from 150.185.138.59: icmp_seq=7 ttl=47 time=7140 ms
[1443758095.072853] 64 bytes from 150.185.138.59: icmp_seq=8 ttl=47 time=6625 ms
[1443758095.625384] 64 bytes from 150.185.138.59: icmp_seq=9 ttl=47 time=6170 ms
[1443758095.701993] 64 bytes from 150.185.138.59: icmp_seq=10 ttl=47 time=5246 m
s

--- 150.185.138.59 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9065ms
rtt min/avg/max/mdev = 5246.154/7292.862/8485.394/958.056 ms, pipe 9
```

Figura 3.3: Ping output

| Parametro | Tipo | Descripcion |
|---|---|---|
| Numero de sondas | Entero | Número de sondas icmp a enviar. |
| Timeout | Float | Tiempo a esperar por una respuesta antes de asumir una sonda como perdida. |
| Intervalo entre sondas | Float | Tiempo entre el envío de cada sonda individual. |

Tabla 3.2: Parametros de la prueba ping

### 3.3.2 Httping

Esta prueba usa el protocolo HTTP para hacer un HEAD Request y obtener el tiempo de respuesta y código de estatus HTTP, a diferencia de la prueba con ping esta no ejecuta un comando externo sino que llama a una función de la biblioteca 'requests' que hace la petición directamente por lo que no es necesario ningún tipo de parsing.

La tabla 3.3.2 muestra los parámetros de la prueba httping,

Esta prueba es útil para monitorear todo tipo de servidores web y posee la ventaja sobre ping de utilizar un protocolo de capa aplicación por lo que da una idea mas precisa de la experiencia del usuario al visitar dicho servicio; a diferencia de ping, el resultado obtenido no solo es la latencia de la red, sino la suma de la latencia de la red y el tiempo de respuesta del servidor, que puede estar sujeto, por ejemplo, al nivel de carga que esté manejando dicho servicio, o a la petición especifica que se esté

| Parametro | Tipo | Descripcion |
|---|---|---|
| Timeout | Float | Tiempo a esperar por una respuesta |
| Path | String | Cadena de caracteres que se adjunta al ip o nombre de dominio del enlace, especialmente útil para probar servicios específicos de una aplicación o servicio web.. |
| Port | Entero | Espeficia el puerto al que se envía la peticion HTTP. |

Tabla 3.3: Parametros de la prueba httping

realizando.

### 3.3.3 Traceroute

Esta prueba ejecuta el comando traceroute para obtener la ruta entre un par de nodos a través de una red ip, llamamos ruta a una secuencia de saltos (hops) que hace un paquete al atravesar un enrutador.

El comando traceroute imprime la ruta como una lista ordenada donde cada linea representa un salto, con su dirección ip, nombre de dominio y latencia, dependiendo del numero de sondas que se estén enviando por salto pueden existir casos en que se obtenga respuesta de mas de una dirección ip, este comportamiento se puede ver en la figura 3.4 en el salto 9 se observa que obtenemos una respuesta de la dirección ip 154.54.31.230 y dos de 154.54.47.154

A partir de la salida de traceroute se debe obtener una estructura de datos que facilite el análisis de la ruta, para esto se usó el modulo traceroute parser.py[1], que extrae la información de la cabecera (ip destino y nombre de dominio), asi como una lista de hops (saltos), cada hop es a su vez una lista de probes (sondas), cada sonda tiene dirección ip, nombre de dominio, rtt y anotaciones; ya que el ip destino es conocido, solo se guarda en el archivo de trazas la lista de saltos en formato json.

El formato esta compuesto de la siguiente manera:

[1] traceroute parser.py es cortesia del proyecto Netalyzr: http://netalyzr.icsi.berkeley.edu

```
jesus@jesus-pc:~$ traceroute 150.185.138.59
traceroute to 150.185.138.59 (150.185.138.59), 30 hops max, 60 byte packets
 1  192.168.1.1 (192.168.1.1)  0.296 ms  0.399 ms  0.504 ms
 2  186.14.23.1 (186.14.23.1)  463.401 ms  463.450 ms  464.369 ms
 3  200.82.134.67 (200.82.134.67)  460.462 ms  461.378 ms  462.339 ms
 4  10.1.232.145 (10.1.232.145)  492.168 ms  493.138 ms  495.226 ms
 5  10.1.230.98 (10.1.230.98)  582.430 ms *  583.399 ms
 6  ro-ccs-bdr-02-TenGig4-1-0.ln.inter.com.ve (10.1.230.62)  496.203 ms  458.475 ms  503.555 ms
 7  tengigabitethernet4-1.asr1.ccs1.gblx.net (64.215.248.93)  505.796 ms  506.888 ms  507.880 ms
 8  te0-0-0-34.ccr21.mia03.atlas.cogentco.com (154.54.12.69)  593.042 ms  591.014 ms  592.035 ms
 9  te4-1.mag01.mia03.atlas.cogentco.com (154.54.31.230)  590.031 ms te7-1.mag01.mia03.atlas.cogentco.com (154.54.47.154)  596.323 ms  593.998 ms
10  * 38.104.95.186 (38.104.95.186)  594.960 ms  570.787 ms
11  pa-us.redclara.net (200.0.204.6)  634.901 ms  637.250 ms *
12  reacciun-pa.redclara.net (200.0.204.150)  761.604 ms  758.751 ms  756.794 ms
13  150.185.255.86 (150.185.255.86)  680.235 ms  677.786 ms  680.704 ms
14  190.168.0.5 (190.168.0.5)  709.765 ms  709.753 ms  709.954 ms
15  150.185.163.248 (150.185.163.248)  710.052 ms  687.120 ms *
16  * 150.185.138.59 (150.185.138.59)  654.827 ms  654.779 ms
```

Figura 3.4: Salida del comando traceroute

```
[
        [
                {
                        "anno": anno_1,
                        "rtt": rtt_1,
                        "ipaddr": ipaddr_1,
                        "name": name_1
                },
                {
                        "anno":  anno_2,
                        "rtt": rtt_2,
                        "ipaddr": ipaddr_2,
                        "name": name_2
                },
                ...
        ],
        ...
]
```

Cada vez que se realiza una prueba con traceroute se anexa al archivo de trazas una entrada con la marca de tiempo de inicio de la prueba, un carácter de separación y luego una cadena en el formato json antes mostrado.

Las parámetros de esta prueba son los siguientes:

| Parametro | Tipo | Descripcion |
|---|---|---|
| Numero de sondas | Entero | Número de sondas a enviar por cada valor de TTL. |
| TTL Maximo | Entero | Numero maximo de saltos antes de asumir que el nodo no es alcanzal |

Tabla 3.4: Parametros de la prueba con traceroute

### 3.3.4 Throughput con Iperf

La prueba de throughput con iperf mide el ancho de banda entre un par de nodos usando el programa Iperf, como se pudo ver en las pruebas anteriores, siempre es necesario que el equipo destino (Tentacle Probe Destination) ofrezca algún tipo de respuesta ya sea en forma de ICMP Echo Reply o HTTP Response, en estos casos no necesariamente hace falta instalar o configurar el equipo destino pues estos ya implementan dichos servicios, sin embargo en el caso de Iperf el usuario que realiza el monitoreo debe asegurarse de mantener una instancia de Iperf en modo servidor para realizar las pruebas, en otras palabras el usuario debe tener acceso o contar con la colaboración explicita de los puntos finales a monitorear.

| Parametro | Tipo | Descripcion |
|---|---|---|
| Probar este enlace | Booleano | Determina si se debe o no ejecutar la prueba para un enlace e |
| Tiempo de transmisión | Entero | Numero de segundos para transmitir datos durante la prueba. |
| Bytes a enviar | String | Cantidad de bytes a enviar durante la prueba, se puede especi |
| Numero de flujos | Entero | Numero de flujos TCP simultáneos a usar durante la prueba. |
| Puerto | Entero | Especifica el puerto en que el servidor está escuchando en el T |

Tabla 3.5: Parametros de la prueba con iperf

## 3.4 Casos de uso

Ya que el monitor de red Tentacle funciona de forma automatizada y todas las acciones de configuración y visualización de los datos recolectados por el se realizan en la aplicación web, solo se tienen cuatro casos de uso para el monitor.

Tabla 3.6: Caso de uso – Iniciar monitor

| ***TM-01*** | *Iniciar monitor* | |
|---|---|---|
| ***Descripción*** | El usuario desea iniciar el monitor de red tentacle. | |
| ***Secuencia normal*** | **Paso** | **Acción** |
| | 1 | El usuario invoca el cliente con el comando "start". |
| | 2 | El cliente crea el proceso daemon y retorna. |
| ***Excepciones*** | **Paso** | **Acción** |
| | 3 | Si el archivo pid existe entonces el cliente muestra un mensaje de error indicando que el monitor ya se esta ejecutando. |

Tabla 3.7: Caso de uso – Detener monitor

| ***TM-02*** | *Detener monitor* | |
|---|---|---|
| ***Descripción*** | El usuario desea detener el monitor que se esta ejecutando en modo daemon. | |
| ***Secuencia normal*** | **Paso** | **Acción** |
| | 1 | El usuario invoca el cliente con el comando "stop". |
| | 2 | El cliente abre el archivo pid y envía la señal SIGTERM al proceso daemon. |
| | 3 | El monitor maneja la excepción deteniendo su ejecución. |
| ***Excepciones*** | **Paso** | **Acción** |
| | 4 | Si el archivo pid no existe, el cliente informa al usuario que el monitor no se esta ejecutando. |

Tabla 3.8: Caso de uso – Reiniciar monitor

| ***TM-03*** | *Reiniciar monitor* | |
|---|---|---|
| ***Descripción*** | El usuario desea reiniciar el monitor que se esta ejecutando en modo daemon. | |
| ***Secuencia normal*** | **Paso** | **Acción** |
| | 1 | El usuario invoca el cliente con el comando "restart". |
| | 2 | El cliente abre el archivo pid y envía la señal SIGTERM al proceso daemon. |
| | 3 | El monitor maneja la excepción deteniendo su ejecución. |
| | 4 | El cliente crea el proceso daemon y retorna. |
| ***Excepciones*** | **Paso** | **Acción** |
| | 5 | Si el archivo pid no existe, el cliente informa al usuario que el monitor no se esta ejecutando. |

Tabla 3.9: Caso de uso – Ver Plan de Monitoreo

| ***TM-04*** | *Ver Plan de Monitoreo* | |
|---|---|---|
| ***Descripción*** | El usuario desea ver el plan de ejecución del monitor. | |
| ***Secuencia normal*** | **Paso** | **Acción** |
| | 1 | El usuario invoca el cliente con el comando "jobs". |
| | 2 | El cliente abre la base de datos y retira la información. |
| | 3 | El cliente muestra por pantalla las tareas planificadas y los enlaces monitoreados y retorna. |
| | **Paso** | **Acción** |
| ***Excepciones*** | 5 | Si la base de datos no se ha creado muestra una lista vacía. |

# Capítulo 4

# Aplicación Web "Optopus Head"

La aplicación web "Octopus Head" hace las veces de interfaz entre el usuario final y sus monitores de red, ademas es el encargado de recolectar los datos obtenidos del monitoreo y crear poderosas visualizaciones interactivas a partir de ellos, mientras los monitores realizan la tarea relativamente sencilla de recolectar datos de las redes a monitorear en Octopus Head se define el plan de monitoreo y se ejecutan las tareas pesadas de análisis de los datos recolectados para computar periodos de actividad continua, mapas de calor, horas y días activos, etc.

## 4.1 Diseño del sistema

Para el desarrollo de esta aplicación web se usó el Framework de desarrollo web Django que implementa un patrón MVC, ya que nuestra aplicación web se encargará de manejar tareas computacionalmente intensas, o de largo tiempo de ejecución, Django se integró con el sistema de procesamiento de tareas distribuido Celery, esto no solo con la finalidad de que el servidor web pueda delegar estas tareas y responder rápidamente al usuario, sino también para permitir una mejor escalabilidad del sistema, que entonces podrá responder a un mayor número de peticiones por unidad de tiempo.

### 4.1.1 Arquitectura

La arquitectura de Octopus Head consiste en cuatro capas, la capa superior o capa de presentación se ejecuta en el navegador del cliente monitor, administrador o quien sea que vea los resultados obtenidos a partir del monitoreo, esta capa se comunica con la capa dos o capa de negocio que es ejecutada por el servidor web que responde a las peticiones de los usuarios, delega tareas a la capa tres y retira y actualiza datos de la capa cuatro, la capa tres es la encargada de ejecutar tareas largas o computacionalmente intensas así como de planificar e iniciar tareas periódicas, la capa cuatro o capa de datos aloja los datos de la aplicación como usuarios, monitores, planes de monitoreo, historiales, datos recolectados de las redes, reportes, etc.

NOTA: Insertar figura de la arquitectura de la maquina servidor

### 4.1.2 Modelo de la base de datos

La base de datos de Octopus Head esencialmente modela un conjunto de monitores junto con sus enlaces monitoreados (tentáculos), los resultados obtenidos para cada tentáculo (o monitor) de cada una de sus pruebas, las pruebas y sus parámetros, el historial de cambios en el plan de monitoreo, entre otros. Ya que Django usa un ORM para manejar la base de datos, desde el punto de vista de la aplicación cada entidad es una clase por lo que es posible aprovechar todas las características de la programación orientada a objetos, como herencia de clases, clases abstractas implementación de métodos específicos a un modelos y sobrecarga de métodos de los padres, a continuación se describen las entidades que se desa modelar:

Cada usuario registrado del sistema está representado por el modelo "User" que viene incluido como parte del framework de Django, este guarda la información mínima necesaria para autenticar al usuario, así como sus grupos y permisos, un usuario autenticado puede estar en uno o mas grupos y puede tener uno o mas permisos, por defecto se tienen tres tipos de usuarios: el usuario normal, el usuario "staff" que puede ingresar la panel de administración del sistema y el superusuario, el superusuario tiene todos los privilegios del sistema, por lo tanto puede manejar otros usuarios (agregando grupos, permisos, cambiando el tipo de otros usuarios, etc) y hacer cambios a cualquier

otro modelo del sistema.

Los usuarios finales del sistema (es decir aquellos que no son staff o superuser) tienen un modelo adicional llamado "UserProfile" (perfil de usuario), para implementar esta funcionalidad podría parecer conveniente sencillamente extender el modelo "User" sin embargo esto interfiere con el mecanismo de autenticacion (que no espera que exista otra tabla de usuarios) de modo que implementar una relación uno-a-uno entre el modelo "UserProfile" y el modelo "User" es preferido. El perfil de usuario aloja los detalles adicionales del usuario monitor como credenciales de Dropbox y códigos de confirmación.

El modelo central de la base de datos es el monitor, que representa un Tentacle Probe Source haciendo pruebas en la red remota. Cada usuario posee uno o mas monitores, el monitor está compuesto por un número de enlaces monitoreados y un plan de monitoreo asi como su horario de sincronizacion y sus detalles específicos como zona horaria, posición geográfica, dirección ip entre otros.

Los tentaculos son un concepto fundamental en el enfoque de monitoreo de este trabajo, cada tentáculo (o enlace monitoreado) consiste de un Tentacle Probe Source (el monitor) y un Tentacle Probe Destination (un nodo monitoreado) y cualquier número de nodos intermedios entre ellos (como enrutadores, proxys, etc), para representar esto usamos el modelo "Link" que guarda el ip del Tentacle Probe Source y otros detalles como posición geográfica, nombre y descripción.

Las pruebas son procedimientos que el usuario puede planificar para que sus monitores ejecuten según una planificación, es a esto a lo que llamamos el 'plan de monitoreo', una prueba consiste en cualquier código python ejecutable, generalmente con el objetivo de obtener algún dato de la red monitoreada, desde el punto de vista de la aplicacion web una prueba se modela como una agregación de parámetros configurables, estos son usados para construir un formulario que permite al usuario editar el plan de monitoreo.

Los resultados de las pruebas también llamados trazas se guardan según las características especificas de cada prueba, no existe ningún limitante a la hora de diseñar modelos para los resultados de las pruebas, sin embargo las pruebas implementadas hasta ahora tienen en común un timestamp (el tiempo de ejecucion

de la prueba) y el tentáculo relacionado a la prueba, sin embargo sería posible guardar trazas que no estén relacionadas a ningún tentáculo (por ejemplo, resultados del sondeo de el número de errores en una interfaz del TPS)

Un "MonitorTest" es una modelo intermedio que representa una prueba planificada para un monitor y sus parámetros, en otras palabras el conjunto de "MonitorTest" para un monitor conforman el plan de monitoreo, este modelo aloja el tiempo inicial de ejecucion de una prueba, el tiempo final, intervalo entre pruebas, y su estado (activa o inactiva).

Ya que el usuario tiene la libertad de modificar el plan de ejecucion en cualquier momento dado, se guarda un historial de los cambios hechos a los "MonitorTest" en una tabla a modo de historial, de esta manera es posible reconstruir con que parámetros estaba siendo ejecutada una prueba en cualquier momento especifico, esto no solo con fines informativos para el usuario, sino también puede ser útil para algoritmos de análisis de los datos que necesiten conocer los parámetros de ejecucion en algún momento especifico.

A los distintos métodos de análisis y visualización los datos que permiten al usuario explorar y obtener información relevante a partir de ellos, las llamamos visualizaciones, computar visualizaciones consiste en retirar los datos seleccionados de la base de datos, pasar estos datos por algún algoritmo de análisis y prepararlos para ser desplegados en alguna forma elegida por el usuario, como mapas de calor, gráficas de barras, mapas, etc.

Las sincronizaciones consisten en recoger los datos dejados por los monitores remotos en la nube y insertarlos a la base de datos, generalmente haciendo algún preprocesamiento o parsing, se guarda un historial de sincronizaciones no sólo como forma de informar al usuario de la cantidad de archivos recolectados y si ocurrieron errores, el mecanismo de sincronizacion depende un cursor que indica al sistema cuales son los archivos nuevos o modificados que deben ser insertados, el mecanismo de sincronizacion se explicará a fondo en la subsección 4.2.2.

Como se vió en la figura **??** el planificador depende de la base de datos para determinar el horario de sincronizaciones, por lo que es necesario un modelo para guardar el horario de sincronizaciones, ya sea este por intervalos o a una hora especifica

del día.

Los reportes son una forma de mostrar y compartir conjuntos de visualizaciones en una sola vista, de esta manera es posible hacer comparaciones de gráficas de distintos tentáculos, o incluso de distintos monitores y ademas compartirlas con usuarios no autenticados haciendo los resultados públicos.

#### Diagrama de clases

#### Diagrama de entidad relación

#### Entidades

#### Relaciones

#### Optimizaciones

### 4.1.3 Flujo de navegación

### 4.1.4 Diseño de Pantallas

## 4.2 Componentes

### 4.2.1 Data Pusher

### 4.2.2 Recolector de datos

### 4.2.3 Algoritmos de análisis

#### Cálculo de Mapas de Calor de RTT

#### Cálculo de horas activas

#### Cálculo de días activos

#### Cálculo de periodos de actividad continúa

## 4.3 Casos de Uso

## 4.4 Caching de gráficas

## 4.5 Pruebas de Rendimiento

# Capítulo 5

# Framework de integración de pruebas

# Capítulo 6

# Conclusiones y Recomendaciones

# Bibliografía


[1] J. F. Kurose and K. W. Ross, *Computer Networking. A Top-Down Approach.* Pearson, 2013.

[2] M. Zennaro, E. Pietrosemoli, J. Mlatho, M. Thodi, and C. Mikeka, "An assessment study on white spaces in malawi using affordable tools," in *Global Humanitarian Technology Conference (GHTC), 2013 IEEE*, (San Jose, CA), pp. 265 – 269, IEEE, 2013.

[3] J. Gomez, M. Porcar, M. Hernandez, and E. Velasquez, "Malawinet network monitor.," tech. rep., Universidad de los Andes, 2015.

[4] B. Vahl, T. Luque, F.H.and Huhn, and C. Sengul, "Network monitoring and debugging through measurement visualization," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium*, (San Francisco, CA), pp. 1 – 6, IEEE, 2012.

[5] S. Cloud, "Pingdom - website monitoring." [Página web en línea] Disponible en https://www.pingdom.com/, 2015.

[6] U. R. B. A.S., "Uptime robot." [Página web en línea]. Disponible en : https://uptimerobot.com/, 2015.

[7] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Transactions on Computer Systems (TOCS)*, vol. 2, no. 4, pp. 277–288, 1984.

[8] I. Grigorik, *High Performance Browser Networking.* O'Reilly, 2013.

[9] S. D. Strowes, "Passively measuring tcp round-trip times," *Communications of the ACM*, vol. 56, no. 10, pp. 57–64, 2013.

[10] C. Demichelis and P. Chimento, "RFC3393: IP Packet Delay Variation Metric for IP Performance Metrics," RFC 3393, RFC Editor, November 2002.

[11] Ookla, "Speedtest.net by ookla." [Página web en línea]. Disponible en : http://www.speedtest.net/, 2014.

[12] M. Dahlin, B. B. V. Chandra, L. Gao, and A. Nayate, "End-to-end wan service availability," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 2, pp. 300–313, 2003.

[13] J. Gettys and K. Nichols, "Bufferbloat: Dark buffers in the internet," *Queue*, vol. 9, no. 11, p. 40, 2011.

[14] S. M. LLC, "Dslreports home: Broadband isp reviews news tools and forums." [Página web en línea]. Disponible en : http://www.dslreports.com/speedtest/, 2015.

[15] N. Ltd, "Thinkbroadband :: Uk broadband speed test." [Página web en línea]. Disponible en : http://www.thinkbroadband.com/speedtest.html, 2015.

[16] R. B. Fragoso, "¿que es big data?," *IBM developerWoks*, 2012.

[17] T. G. Flu and D. T. Team, "Google flu trends." [Página web en línea]. Disponible en : https://www.google.org/flutrends/about/, 2012.

[18] Pervasifm, "Data visualization." [Página web en línea]. Disponible en : http://www.pervasif.com/index.php/news-a-event/capabilities/data-visualization, 2012.

[19] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[20] E. B. Pantoja, "El patrón de diseño modelo-vista-controlador (mvc) y su implementación en java swing," *Acta Nova*, vol. 2, no. 4, p. 493, 2004.

[21] Oracle, "Mysql 5.7 :: Reference manual 1.3.1 what is mysql?." [Página web en línea]. Disponible en : http://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html, 2015.

[22] S. W. Ambler, "Mapping objects to relational databases: O/r mapping in detail." [Página web en línea]. Disponible en : http://www.agiledata.org/essays/mappingObjects.html, 2013.

[23] I. Dropbox, "Acerca de dropbox." [Página web en línea]. Disponible en : https://www.dropbox.com/about, 2015.

[24] I. Poese, S. Uhlig, M. A. Kaafar, B. Donnet, and B. Gueye, "Ip geolocation databases: Unreliable?," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 53–56, 2011.

[25] A. Grönholm, "Advanced python scheduler - apscheduler 3.1.0.dev1 documentation." [Página web en línea]. Disponible en : https://apscheduler.readthedocs.org/en/latest/, 2015.