Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE-0117 Programación bajo plataformas abiertas
I ciclo 2016

# Proyecto 2

# *Manejador de redes inalámbricas inteligente y programable*

Aarón Sibaja Villalobos, B56891
Andrés Vargas Salguero, B06670

Profesor: Federico Ruiz Ugalde

3 de julio del 2016

# Índice

# Índice de figuras

# 1. Descripción:

El presente proyecto se plantea confeccionar un software administración de redes inalámbricas que posea un algoritmo o demonio que sea capaz de diagnosticar y evaluar la mejor red disponible, según los parámetros que sean de mayor peso para el usuario, conectarse a esta, y que constantemente siga evaluando con el fin de que las condiciones de red sean siempre las más óptimas.

# 2. Objetivo General:

Crear un gestor de redes inalámbricas que posea características de diagnóstico que evalúen constantemente redes disponibles y se conecte a la que posea las mejores condiciones, según las preferencias propias del usuario.

# 3. Objetivos específicos:

1. Diseñar una función de costo de red (tomar en cuenta velocidad de bajada, velocidad de subida, calidad de señal, estabilidad y latencia).

2. Construir un demonio que evalúe la red más adecuada para conectarse y se conecte a ella mientras siga evaluando las redes de manera periódica.

3. Construir un programa en consola que se comunique con el demonio para obligarlo a diagnosticar las redes automáticamente.

4. Integrar el programa anterior al sistema de eventos del kernel de Linux(Udev) para que cuando pierda conexión intente nuevamente.

5. Construir un demonio que calcule periódicamente todos lo costos mencionados por aparte y en conjunto con la hora, el día y la red particular y almacene los datos en un archivo.

6. Construir un programa que tome el archivo anterior y utilizando SVM genere un predictor.

7. Integrar este predictor al demonio del segundo objetivo.

# 4. Justificación

Con el paso del tiempo y con la introducción de nuevas tecnologías de comunicación, nos encontramos en un mundo donde las redes juegan un papel casi que vital dentro del desarrollo de nuestras vidas. Cada día son más las personas que necesitan de alguna red para poder realizar ciertas actividades; desde realizar efectivamente trabajos hasta la comunicación con otras personas.

La mayoría del tiempo las personas recuren a conexiónes inalambricas por su conveniencia, comodidad y fácil acceso. Sin embargo, cada persona tiene sus gustos y prioridades al respecto, que muchas veces depender de las actividades que necesiten llevar a cabo. Algunos prefieren la estabilidad de conexión cualquier cosa. Otras pesonas no consideran importante sacrificar este apartado, siempre y cuando se cuente con velocidades máximas de carga y descarga. También encontramos algunos que utilizan ciertos servicios de voz o streaming y requieren conexiones con baja latencia.

Normalmente los manejadores de redes disponibles para los sistemas operativos vienen con estos paramentros de conexión establecidos por defecto por el desarrollador. Teniendo esto en cuenta, es de esperarse que las preferencias de los usuarios no sean atendidas de una manera completa, ya que lo que es primordial para algunos, no necesariamente lo será para otros.

Justamente sobre este problema es que se está planteando la confección de un gestor de redes inalámbricas que sea flexible y programable para que cada usuario pueda ajustar según lo que desea o necesite. Que sea posible establecer listas de prioridades y que estas se diagnostiquen constantemente para brindarle la mejor experiencia de anejo de redes que sea posible.

# 5. Metodología

Inicialmente el proyecto se planteó de una manera diferente a como fue desarrollado, y por eso se variaron de cierta manera los objetivos. Esto fue debido a varias razones de peso, decisiones personales, por simplicidad o complicaciones durante el desarrollo.

Es importante que se aclare el motivo por el cual se abandonó el objetivo de SVM (Machine Learning). Como se explica más adelante se decidió hacer una función de pesos para la evaluación de la mejor red para el usuario, y en dicha función se aplican promedios y otras funciones que en teoría representan la idea de hacer Machine Learning. Sin embargo cabe resaltar que la implementacion de Support Vector Machine, puede lograr que el programa prediga la mejor situación (aproximación) cuando los datos se salgan de lo esperado o estén muy lejos de los datos obtenidos en análisis previos.

## 5.1. Plan inicial

Para el desarrollo del proyecto se pretendía dividir el trabajo en tres partes principales. Primero diseñar una función para evaluar los costos de la red, las caraterísticas necesarias que describen la eficiencia de una red. Un demonio ayudará a calcular periódicamente estos costos con frecuencias suminstradas por el usuario, día, semana, hora.

Después de esto construir un segundo demonio para que evalué constantemente la red más adecuada para conectarse de manera frecuente. Además de aplicar un algoritmo de machine learning llamado Support Vector Machine, integrado por defecto en las librerías de Python

Por último se pretendía integrar los módulos anteriores con el sistema de eventos del kernel del sistema operativo GNU/Linux(Udev) para que cuando se pierda conexión intente evaluar y conectar nuevamente.

## 5.2. Plan final

Se optó por cambiar un poco la manera de desarrollo del proyecto por varias razones enumeradas anteriormente, pero siempre manteniendo la misma idea general de un manejador inteligente y programable que funcione bajo la ejecución de dos demonios diferentes.

Después de un análisis denso, se decidió tomar en cuenta sólo tres variables de red con el fin de que la elaboración del programa se viera simplificada. Esta variables escogidas fueron: velocidad de descarga (bajada), velocidad de carga (subida), e intensidad de señal. Elaborar el software dejando de lado otras variables, no irrumpe el objetivo de tener un producto final funcional. Además, la adicción de estas variables mencionadas no es una tarea difícil.

Bajo la misma lógica de simplicidad, se optó por considerar solo redes inalámbricas abiertas o sin protocolos de seguridad.

Primeramente se desarrollaron los demonios. El primero capaz de escanear las redes al alcance, y para cada una extraer la SSID, la intensidad de la señal en *dBm* y si esta una red con seguridad habilitada o no. Seguidamente, intenta conectarse a la primera red disponible que no posea protocolos de seguridad, y sin retraso alguno ejecuta un método de análisis de velocidades (carga y descarga). Este demonio es capaz de guardar todos estos datos. El segundo demonio recibe los datos del primero, y mediante librerías especiales para escritura de datos, almacena todos los datos respectivos a cada SSID en un archivo que luego va a ser leído por el programa.

Finalmente se desarrolló una función de pesos de red, que tiene como entrada un archivo sencillo de configuración, que cada usuario puede editar a su gusto personal. La idea es que a cada variable de red se le asigne un peso que indique cuál de estas es más importante para el usuario, en una escala del uno al cinco. Los datos recolectados y almacenados por los demonios vienen a ser la segunda entrada de esta función. En teoría, la red "más optima"sería la que posea una función de pesos mayor.

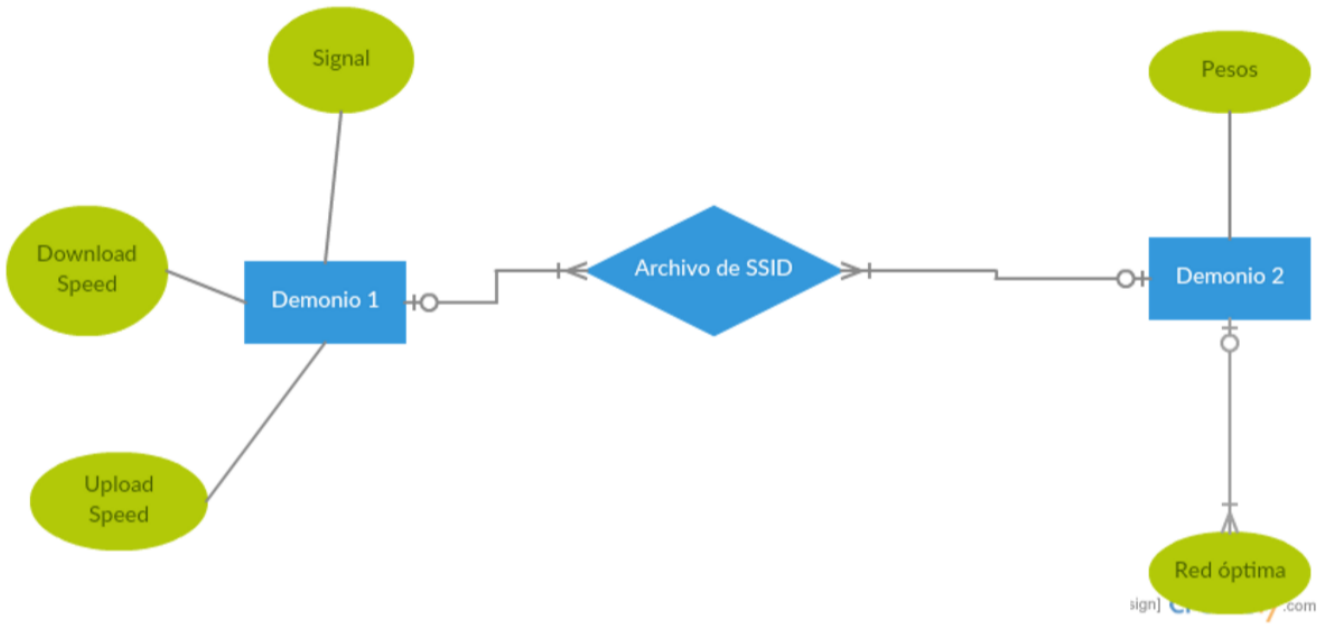


Figura 1: Metodología de desarrollo final del proyecto

# 6. Marco Teórico

## 6.1. Service Set Identifier (SSID)

Este indicador es sencillamente un nombre incluido en todos los paquetes de una red inalámbrica que los identifica como parte de esa red. Consiste en un máximo de 32 caracteres, que por conveniencia se utilizan caracteres alfanuméricos, aunque el estándar no lo especifica explícitamente. Todo dispositivo inalámbrico que intenta comunicarse con cualquier otro dentro de una red debe compartir este indicador. Generalmente, se puede comprender como el nombre que un administrador le da a una red para ser identificada.

## 6.2. Received Signal Strength Indicator (RSSI)

El indicador de fuerza de la señal recibida es una escala de referencia, en relación a 1 $mW$ para medir el nivel de potencia de señal de una red inalámbrica recibida por un dispositivo bajo las variantes del estándar IEEE 802.11. La escala tiene al valor cero como inicio (0 $dBm = 1\ mW$). Generalmente en redes inalámbricas, la escala se expresa dentro de valores negativos; cuanto más negativo, menor es la intensidad de la señal. Es importante indicar que lo que se mide es intensidad recibida, no calidad de señal; esta última se determina contrastando la intensidad con respecto al ruido (interferencias, etc). A continuación, en la Figura 1, se detallan los valores de la escala y sus intensidades correspondientes.

| Valor (dBm) | Interpretación de señal |
|---|---|
| 0 | Señal “ideal”, utópica |
| -10 a -30 | Señal excelente |
| -40 a -60 | Señal estable |
| -60 | Enlace bueno |
| -70 | Enlace normal/medio |
| -80 | Señal mínima |

Figura 2: Escala RSSI de intensidad de señal.

## 6.3. Velocidades de Internet (Download/Upload)

Las velocidades de internet de carga y descarga son una medida de tasa de bits. Estas definen el número de bits que se transmiten por unidad de tiempo a través de un sistema de transmisión digital o entre dos dispositivos digitales. Esta medida es afectada directamente por varios factores como la topología de red, número de usuarios conectados, aplicaciones

que utilizan ancho de banda, el servidor, congestión de la red y las especificaciones técnicas de los dispositivo (tajeta de red, procesador, etc.).

La unidad con que el Sistema Internacional de Unidades expresa la tasa de bits es el bit por segundo (*bit/s, b/s, bps*). La b debe escribirse siempre en minúscula, para impedir la confusión con byte por segundo (*B/s*). Para convertir de bytes/s a bits/s, basta simplemente multiplicar por "8ƶ viceversa. Actualmente como las velocidades de Internet y capacidades de ancho de banda han ido creciendo considerablemente, la unidad *bit/s* ha pasado a un segundo plano, y se han empezado a utilizar múltiplos de la misma. En la Figura 2 se explican los diferentes múltiplos de la unidad *bit/s*.

| Unidad | Otras denotaciones | Significado |
|---|---|---|
| kbit/s | kbps, kb/s | mil bits por segundo |
| Mbit/s | Mbps, Mb/s | un millon de bits por segundo |
| Gbit/s | Gbps, Gb/s | mil millones de bits por segundo |
| B/s | byte/s | 8 bits por segundo |
| kB/s | kilobyte/s | mil bytes por segundo |
| MB/s | megabyte/s | un millon de bytes por segundo |
| GB/s | gigabyte/s | mil millones de bytes por segundo |

Figura 3: Múltiplos de la unidad bits por segundo.

## 6.4. Materiales de apoyo. Librerías de python.

El lenguaje de programación Python cuenta con gran cantidad de librerias que vienen incluidas por defecto o que fueron desarrolladas por una comunidad de usuarios y publicadas para el uso general. Utilizamos varias de ellas para la elaboración de este software, tanto externas como internas. Entiéndase externas como aquellas librerías que no vienen incluidas por defecto en python y que necesitan de una instalación previa para ser implementadas de buena manera. Las librerías incluidas en python que fueron utilizadas son:

- **Ejecución de comandos de consola en python:** os, commands
- **Análisis de la salidad de un comando de consola:** subproccess
- **Funciones útiles de tiempo:** time
- **Descrubrimiento de patrones en texto:** re o RegEx (Regular Expressions)

## 6.5. Librerias Externas

### 6.5.1. Python-daemon 2.1.1

Es descrita por los desarroladores como una librería para implementar un demonio "bien portado.[en] sistemas operativos tipo Unix. El proceso "demonizado"se ejecuta sin que se vea detenido por alguna inconsistencia, y hasta que el usuario decida detenerlo. Además no interviene con ningún otro proceso importante.

### 6.5.2. YAML (pyYAML)

Esta libreria implementa funciones y métodos para la correcta escritura y lectura de archivos YAML (.yml). Es un formato de serialización de datos, bastante ligero, que es legible por humanos y que fue desarrollado específicamente para python en el año 2001. YAML representa un acrónimo recursivo que significa "YAML Ain't Another Markup Language.º "YAML no es otro lenguaje de marcado.[en] castellano.

Fue creado bajo la creencia de que todos los datos pueden ser representados adecuadamente como combinaciones de listas, hashes (mapeos) y datos escalares (valores simples). La sintaxis es relativamente sencilla y fue diseñada teniendo en cuenta que fuera muy legible pero que a la vez fuese fácilmente "mapeable.[a] los tipos de datos más comunes en la mayoría de los lenguajes de alto nivel.

### 6.5.3. Speedtest-cli 0.3.4

Libería sencilla, aún en desarrollo, que utiliza todo la estructura de la conocida página web *www.speedtest.net* de análisis de velocidades de Internet, para mostrar los resultados de la prueba de velocidades en consola. Para hacer uso de la librería nada mas basta con ejecutar el comando *speedtest* o *speedtest-cli* en una terminal o mediante python.

### 6.5.4. Resumen

| Libreria | Versión | Descripcion |
|---|---|---|
| python-daemon | 2.1.1 | Demonización de procesos |
| pyYAML | 2.7 | Escritura/lectura de archivos .yml |
| Speedtest-cli | 0.3.4 | speedtest.net en python |

Figura 4: Liberías externas a python utilizadas

# 7. Estado del arte

## 7.1. Gnome Network Manager (applet

Es la herramienta estándar de configuracion de redes de Linux. Fue desarrollado por GNOME Project y soporta un largo rango de configuraciones de red, desde computadoras de escritorio, servidores y hasta dispositivos móviles. Es compatible con gran cantidad de entornos de escritorios en el mercado actual (GNOME, KDE, Xfce) y acepta la integración de extensiones y plugins. Fue hecho para estar bien integrado a muchas distribuciones diferentes de GNU/Linux; entre ellas Fedora, Debian, Ubuntu, CentOS, SUSE, etc. Presenta integración con la linea de comandos (terminal) de Linux y una amplia interfaz gráfica de edición de conexiones integrada a los distintos entornos de escritorios para los que es compatible.

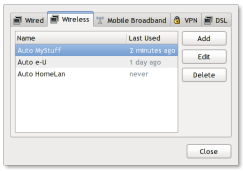


Figura 5: GNOME Network Manager applet.

## 7.2. Wicd Network Manager

Conocido manejador de conexiones de red que puede manejar tanto conexiones alámbricas como inalámbricas, y se ve como una alternativa al Network Manager de GNOME. Está escrito en python y GTK+, y tiene soporte para varios entornos de escritorio, entre ellos Xfce y KDE. Hay maneras de que Wicd sea ejecutado desde la terminal, sin el requerimiento de sesiones gráficas ni paneles de tareas. Cuenta con una sencilla interfaz gráfica para conexiones y configuraciones. Sin embargo, muchos usuarios reportan que el establecimiento de conexiones de red es un poco lento y esporádicas desconexiones inesperadas.


Figura 6: Wicd Network Manager.

# 8. Manual de Usuario

Siga las siguientes indicaciones para ejecutar de manera correcta los demonios en su computadora:

- Lo primero, y probablemente lo más importante por realizar es desactivar el Network-Manager applet que este corriendo el sistema. En nuestro caso sería el nm-applet de GNOME, que viene incluido por defecto en GNOME, o él que se instaló desde un principio si se siguió el manual de instalación de Debian de este curso. Para ello abrimos una terminal y escribimos el siguiente comando:

  ***sudo /etc/init.d/network-manager stop***

- Suponiendo que ya se cuenta con Git instalado y configurado en el sistema, cambie a un directorio óptimo para clonar nuestro repositorio de GitHub. Luego escriba lo siguiente en la terminal para obtener todo el código fuente:

  ***git clone https://github.com/asvaaron/NetworkManager.git***

- Asegúrese que los archivos tengan los permisos respectivos de lectura, escritura y ejecución. De no contar con permisos, puede hacerlo usted mismo mediante el comando de consola ***chmod***.

- Use algun editor de texto (Gedit) y abra el archivo ***pesos.yml***. Establezca el peso que va a tener cada variable de red con una escala del 1 al 5, siendo 1 el peso menor y 5 el peso mayor.

- Los demonios se deben ejecutan y se detienen desde una terminal en modo root, con las indicaciones ***start*** y ***stop***. Por ejemplo, al ejecutar el demonio 1, las indicaciones de consola para ejecutar y detener son respectivamente

  ***./Daemon1.py start , ./Daemon1.py stop***

  ***Nota:*** Es bien sabido que no es seguro identificarse como root en una sesion gráfica ordinaria, pero al tratarse de un demonio que trabajan directamente con el sistema operativo o se integran a este, no habrán brechas de seguridad en el sistema, ademas que los demonios necesitan acceso sudo para cumplir con las funciones de conexión y análisis de red.

- Ejecute por aparte cada demonio y luego corra el ***main.py*** después de algún tiempo para que se le calcule la función de pesos a cada red analizada. El programa con este resultado le sugiere cual es la red más óptima según el peso que le dio inicialmente a cada variable.

- Es importante que tenga al menos dos redes abiertas, ya que por el momento el demonio 1 solo analiza redes sin protocolos de seguridad.

# 9. Conclusiones

Fue posible generar un software que evalúa la mayor cantidad de redes abiertas el cual posea características de diagnóstico que ayude a predecir las redes más óptimas dependiendo de las necesidades del usuario. La función de costos de red le asigna pesos a las características de la red con esto se pretende crear un software más especializado.

El primer demonio cumple con el objetivo planteado desde un inicio, sin embargo se debe de mejorar en el tiempo que transcurre entre la evaluación y la próxima ejecución puesto que esto no le permite seguir trabajando en la red actual al usuario. Se debe evaluar si la red está siendo utilizado y asignarle una prioridad de uso, de esta forma tener en cuenta si se decide continuar la evaluación de las siguientes redes y no perjudicar las tareas que requieren conectarse a una red.

Para medir la latencia de una red y su eficiencia es posible comunicar el software con el sistema de eventos del kernel de GNU/Linux llamado (Udev), esto con el propósito de tener mejores resultados y datos más exactos sobre dicha red, pues se requiere una red que este disponible para cumplir con las mejores características dependiendo de la preferencias antes evaluadas.

Establecer un Machine Learning puede optimizar los resultados de la red de esta forma eliminando datos extremos que pueden afectar el promedio de los datos, además es posible interpolar algunos datos para predecir la eficiencia de la red dependiendo del día de la semana y la hora.

Por último, crear una interfaz gráfica siempre genera una impresión positiva para el usuario, además de que facilita el manejo de archivos y elimina posibles errores en la ejecución del programa, pues se controla un tipo de estructura establecida para que el software funcione correctamente. Es una mejora que se pretende desarrollar en el futuro y generar una aplicación robusta y funcional en cualquier sistema operativo que este basado en el kernel GNU/Linux.

# 10. Anexos

## 10.1. Cronograma

Cronograma de actividades del proyecto:

| | **Fecha** | **Actividad** | **Participantes/Notas** |
|---|---|---|---|
| 1 | 26 mayo | Elaboración de la prepropuesta de proyecto para revisión | Borrador para revisión |
| 2 | 4 Junio | Presentación de la propuesta de proyecto revisado | Presentación oral |
| 3 | 30 mayo - 4 junio | Función de costos de Red | Andrés Vargas |
| 4 | 5 al 10 junio | Demonio de conexión y diagnostico | Aarón Sibaja |
| 5 | 11 al 15 junio | Integración a eventos de Linux | Aaŕon Sibaja |
| 6 | 16 al 19 junio | Demonio generador de archivo de costos | Andŕes Vargas |
| 7 | 20 al 24 junio | Demonio de conexión y diagnostico; Predictor con SVM | Aarón Sibaja |
| 8 | 25 al 27 junio | Integración final del programa | Aarón Sibaja Andrés Vargas |
| 9 | 25 al 27 junio | Período de prubeas y montaje de la presentación final | Video de demostración |
| 10 | 27 junio - 1 julio | Presentación final con prueba de funcionalidad | |

# Referencias

[1] Python-daemon 2.1.1 library for python *https://pypi.python.org/pypi/python-daemon/*

[2] Speedtest-cli 0.3.4 library for python *https://pypi.python.org/pypi/speedtest-cli/*, GitHub: *https://github.com/sivel/speedtest-cli.git*

[3] Documentación ofical de PyYAML extraída del sitio web: *http://pyyaml.org/wiki/PyYAMLDocumentation* el 15 de junio de 2016