

IES Palas Atenea

Proyecto de Investigación Bachillerato de excelencia

# **Programación, Redes y Código Libre**

---

*David Davó*

Tutor  
Julio Sánchez

7 de septiembre de 2016

# Índice general

<b>0. Introducción??</b>	<b>1</b>
<b>1. Programación y código libre</b>	<b>2</b>
1.1. Herramientas . . . . .	2
1.1.1. GNU/Linux . . . . .	2
1.1.2. Git y Github . . . . .	2
1.1.3. LaTeX . . . . .	4
1.1.4. Python . . . . .	4
1.1.5. Gtk+ . . . . .	4
1.1.6. Atom . . . . .	5
1.1.7. Wireshark . . . . .	5
<b>2. Redes Informáticas</b>	<b>6</b>
2.1. Capas de Red/Modelo OSI . . . . .	6
2.2. Topologías de red . . . . .	6
2.2.1. Clasificación de las topologías de red . . . . .	7
2.2.2. Nodos de una red . . . . .	8
2.2.3. Enlaces de red . . . . .	9
2.3. Paquetes de red . . . . .	10
2.3.1. Ejemplo: Paquete de red . . . . .	10
2.4. Protocolos . . . . .	10
2.4.1. Familia de protocolos de internet . . . . .	11
2.5. Seguridad de redes . . . . .	13
2.5.1. Tipos de ataques . . . . .	13
2.5.2. Contramedidas . . . . .	14
<b>3. El simulador de redes</b>	<b>16</b>
3.1. Instalación . . . . .	16
3.1.1. Ubuntu / Debian . . . . .	16
3.1.2. Arch Linux . . . . .	16
3.1.3. Ejecución manual / instalación portable . . . . .	16
3.2. Uso del programa . . . . .	17
<b>Glosario y acrónimos</b>	<b>18</b>
<b>A. Unidades de transferencia de datos</b>	<b>21</b>
<b>B. Código del programa</b>	<b>22</b>
B.1. Main.py . . . . .	22
B.2. Modules/logmod.py . . . . .	53
B.3. Modules/save.py . . . . .	54

# Capítulo 0

## Introducción??

Internet, El Internet o La Internet consiste en una caja negra misteriosa con un indicador LED rojo que parpadea y contiene todo el poder del mundo en su interior. El internet no tiene cables, es muy ligero e increíblemente pequeño. Internet se encuentra en lo alto del Big Ben (donde hay mejor recepción) y antes de sacarlo de allí debe ser desmagnetizado por un Gran Maestro del Internet (*Elders of the Internet*) como Stephen Hawking, Linus Torvalds o Richard Stallman. [5]

# Capítulo 1

## Programación y código libre

### Propuesta

El objetivo es el desarrollo de un software programado en Python de código libre con el que los alumnos puedan aprender tanto sobre redes como de programación en Python.

### 1.1. Herramientas

El programa ha sido creado con herramientas de software libre. Según la Free Software Foundation “«Software libre» es el software que respeta la libertad de los usuarios y la comunidad. A grandes rasgos, significa que los usuarios tienen la libertad de ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software. Es decir, el «software libre» es una cuestión de libertad, no de precio. Para entender el concepto, piense en «libre» como en «libre expresión», no como en «barra libre». En inglés a veces decimos «libre software», en lugar de «free software», para mostrar que no queremos decir que es gratuito.” –[4]

Todas las herramientas citadas a continuación, son o están basadas en Software Libre.

#### 1.1.1. GNU/Linux

También llamado incorrectamente sólo Linux, es una manera de llamar al Sistema Operativo (OS) combinación del kernel Linux (Basado en Unix) y el OS *GNU's Not Unix* (GNU no es Unix) (GNU), ambos softwares son libres y de código abierto. Está basado en, y es uno de los ejemplos de, el código libre. Es el sistema operativo más utilizado, ya que la mayoría de los servidores lo usan, y además, otros sistemas operativos como Android están basado en éste.

### Distros

Son las distintas distribuciones de software de GNU/Linux. Es decir, un conjunto de software preconfigurado y compilado formado por el Sistema Operativo GNU, el kernel de Linux y otros tantos paquetes, dependiendo de los usuarios a los que esté dirigida esta. Pueden crearse con el soporte de una empresa; como Ubuntu (Amazon Canonical Ltd.), openSUSE (Novell) o Fedora (Red Hat); y otras mantenidas por comunidades como Debian, Gentoo o Arch Linux.

He usado dos distros diferentes. Una llamada Arch Linux, que es *rolling release* (No tiene “versiones”, sino que siempre se va actualizando con los últimos paquetes disponibles, por lo que siempre está actualizado) y otra es Ubuntu 16, basado en Debian, por lo que está bastante menos actualizado y se han tenido que hacer correcciones en el programa para que pueda funcionar con versiones más antiguas de las dependencias.

#### 1.1.2. Git y Github

Git es un software diseñado por Linus Torvalds con el que puedes crear un Sistema de Control de Versiones o VCS (*Version Control System*). Este programa te permite de forma sencilla volver a una versión o *commit* anterior del programa, así como enviarlas a un repositorio remoto

e incluso publicarlas en línea. Su punto fuerte son las *branches* o “ramificaciones” del código, haciendo que la rama *master* (principal) siempre pueda ser usada. Para ello creamos una nueva rama para cada nueva funcionalidad del programa. La implementación del nuevo código a otra rama se denomina *merge*. Otra de las funcionalidades que implementa es *clone*, que te permite descargar un proyecto si tienes la URL del repositorio git.

Para usar Git, se suele recomendar seguir un *Git workflow* o flujo de trabajo de Git. El más común es el basado en 4 nuevas ramas, a parte de *master*.

**Develop:** es la rama de desarrollo. Se van aplicando las nuevas funcionalidades a esta rama, para luego convergerlas en la rama Release que se va a publicar.

**Release:** una vez hayamos terminado en la rama de desarrollo, se converge Develop con Release y se procede a solucionar los bugs que se vayan descubriendo. Cuando se hayan solucionado todos los bugs y la siguiente versión del programa esté disponible para el público, se hace merge en Develop y en Master, además de aplicarle al commit una etiqueta con el nombre de la versión. (2.2.1, por ejemplo).

**Hotfix:** Es una rama dedicada a solventar los bugs que un usuario descubra en una versión ya lanzada de la aplicación. Cuando un usuario descubre un bug, se crea una nueva rama a partir de la última versión de master, se soluciona el bug en esa rama y luego se vuelve a hacer merge en master.

**Feature <x>:** Donde <x> el nombre de la funcionalidad. Es una rama dedicada a una nueva funcionalidad, se crea a partir de Develop, y una vez terminada, se hace merge en Develop de nuevo. Una cosa nueva.

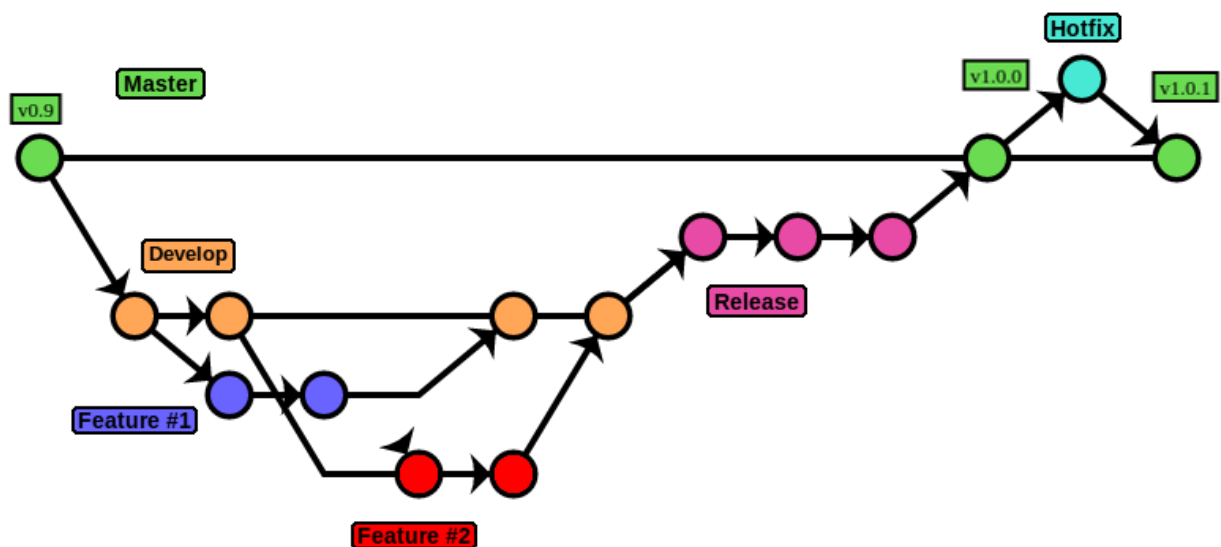


Figura 1.1: *Gitflow* o flujo de trabajo de Git

GitHub es una plataforma de desarrollo colaborativo que te permite alojar tus repositorios Git. Su uso es gratuito si el código almacenado es público. Además, te permite tener una wiki y una página web para tu proyecto, junto a otras funciones. Una de sus funciones estrella es la visualización online del repositorio, con la que cualquier persona tiene acceso al código y los archivos antes de descargarlos. Otra función útil es el apartado de *Issues*, en el que los usuarios de tu código pueden reportar los bugs del programa o aportar nuevas ideas en forma de "foro". Tanto el programa como este documento están disponibles en GitHub en los siguientes enlaces. <https://github.com/daviddavo/InvProy> y <https://github.com/daviddavo/InvProy-tex>

### 1.1.3. LaTeX

$\text{\LaTeX}$  o, en texto plano, LaTeX, pronunciado con la letra griega Ji (X), es un software libre orientado a la creación de textos escritos comparable a la calidad tipográfica de las editoriales. Mediante la importación de paquetes y comandos o macros se puede dar formato al texto al igual que con cualquier otro editor, exportándolo posteriormente a PostScript o PDF. Está orientado a documentos técnicos y científicos por su facilidad a la hora de incluir fórmulas o código e importar paquetes que cumplan tus necesidades. No es un procesador de textos, pues está más enfocado en el contenido del documento que en la apariencia de éste. El código del documento puede ser editado con cualquier editor de texto plano como *nano* o *emacs*, aunque he usado una IDE llamada **texmaker**.

### 1.1.4. Python

Es un lenguaje de programación interpretado (sólo traducen el programa a código máquina cuando se debe ejecutar esa parte del código, por lo que no hace falta compilarlo) que destaca por pretender una sintaxis más legible que la de el resto de lenguajes. Soporta tanto programación imperativa como programación orientada a objetos. Usa variables dinámicas, es multiplataforma, y, además, es de código abierto, lo que permite distribuir el programa en Windows al distribuir los binarios de Python junto a él. En este caso, la versión de Python usada es la 3.4 en adelante.

### 1.1.5. Gtk+

Es un conjunto de bibliotecas o librerías (conjunto de funciones y clases ya definidas preparadas para el uso de los programadores) desarrollado por la GNOME foundation destinado a la creación de GUIs (Interfaz Gráfica de Usuario), también, al igual que Linux forma parte del proyecto GNU.

Contiene las bibliotecas de GTK, GDK, ATK, Glib, Pango y Cairo; de las que he usado fundamentalmente GTK para crear la interfaz principal del programa; GDK al usarlo como intermediario entre los gráficos de bajo nivel y alto nivel y Cairo para la creación de algunos de los elementos gráficos del programa.

Al usar este conjunto de librerías, he conseguido que sólo sea necesario descargar una dependencia del programa, que además suele venir instalada en la mayoría de distros de Linux. Por ejemplo en una instalación limpia de Ubuntu 16 (sin descargar paquetes adicionales) el programa funciona perfectamente. Para usarlo en Python se ha tenido que importar la librería de PyGtk, que también suele venir incluida en la distribución.

### 1.1.6. Atom

Atom es un editor de código multiplataforma con soporte para plugins escrito en Node.js, también tiene soporte para Git. También es un programa de código libre haciendo uso de la licencia MIT.

### 1.1.7. Wireshark

Wireshark es un *packet sniffer* o analizador de paquetes. Te muestra los paquetes de red reales enviados y recibidos por una tarjeta de red, lo que facilita la creación del simulador de redes. También te separa las distintas partes de la encapsulación del paquete, además te permite añadir distintos filtros para las distintas capas.

# Capítulo 2

## Redes Informáticas

### Historia

Internet, tal y como lo conocemos ahora, haciendo uso de IPv6, HTML5, CSS3 no existía hasta hace poco, pero el desarrollo de éste transcurre desde los años 60. En 1961 se publican los primeros artículos de Conmutación de paquetes

### 2.1. Capas de Red/Modelo OSI

El modelo OSI (*Open Systems Interconnection* (Interconexión de Sistemas Abiertos)) es un modelo de referencia para redes basado en capas de abstracción. El objetivo del modelo OSI es conseguir la interoperabilidad entre sistemas con la protocolos estandarizados. Fue creado en 1980 por la ISO (*International Organization for Standardization*). No es considerado una arquitectura de red porque los protocolos no forman parte del modelo, sino son entidades de distintas normativas internacionales.

Capa	PDU <sup>1</sup>	Función	Ejemplos
1. Física	Bit	Transmisión y recepción de bits físicos sobre un medio físico (topología de red)	RJ45, IEEE 802.11, etc.
2. Data Link	Frame	Transmisión segura de <i>frames</i> entre dos nodos conectados por una capa física.	Ethernet, 802.11, etc...
3. Red	Paquete	Estructurar y administrar una red multinode. Incluye enrutamiento, control de tráfico, y asignación de direcciones	IPv4, IPv6, ICMP...
4. Transporte	Datagrama(UDP) Segmento(TCP)	Transmisión de segmentos de datos entre los puntos de una red, incluyendo ACK	TCP, UDP...
5. Sesión	Datos	Administración de sesiones de comunicación, como intercambio continuo de información entre dos nodos.	SSH, RPC, PAP...
6. Presentación	Datos	Translación de datos entre un servicio de red y una aplicación. Incluye comprensión, encriptación/decriptación, y codificación de caracteres.	MIME, TLS
7. Aplicación	Datos	APIs de alto nivel, incluyendo recursos compartidos y acceso remoto de archivos	HTTP, FTP, SMTP...

### 2.2. Topologías de red

La topología de red es la configuración de los elementos que componen una red. Puede ser representada lógica o físicamente. La topología lógica puede ser igual en dos redes, aunque su topología física (distancia entre conexiones, tipo de señales...) pueda ser distinta. Se distinguen

<sup>1</sup>Protocol Data Unit o Unidad de Datos de Protocolo.



dos elementos: los nodos (Ordenadores, switches, etc.) y los enlaces (medio de transmisión de los datos).

### 2.2.1. Clasificación de las topologías de red

Se distinguen ocho tipos de topologías de red: [1]

**Punto a punto:** conexión directa entre los dos puntos de la red. También es conocida como *P2P* (*Peer to Peer*).

**Estrella:** cada host se conecta a un hub central con una conexión P2P. Cada nodo está conectado a un nodo central que puede ser un router, hub o switch.

**Bus:** cada nodo está conectado a un sólo cable. Una señal de un dispositivo viaja en ambos sentidos por el cable hasta que encuentra el destino deseado.

**Anillo:** es una topología en bus pero con los extremos conectados. Los datos atraviesan el anillo en una única dirección y van atravesando cada uno de los nodos, por lo que si uno de ellos no funciona, la red tampoco.

**Malla:** se pueden distinguir dos tipos: completamente conectados, en la que todos los nodos están conectados entre ellos y parcialmente conectados, en la que algunos nodos pueden estar conectados punto a punto y otros pueden tener varias conexiones.

**Híbrida:** combinan dos o más topologías. La más famosa es la topología de **árbol**, en la que se conectan varias topologías de estrella mediante bus.

**Cadena:** se conecta cada ordenador en serie con el siguiente. Cada ordenador repite el mensaje al siguiente ordenador si éste no es su destino. Si se cierra el circuito se crea una topología en anillo, mientras que si se deja abierto se denomina topología lineal.

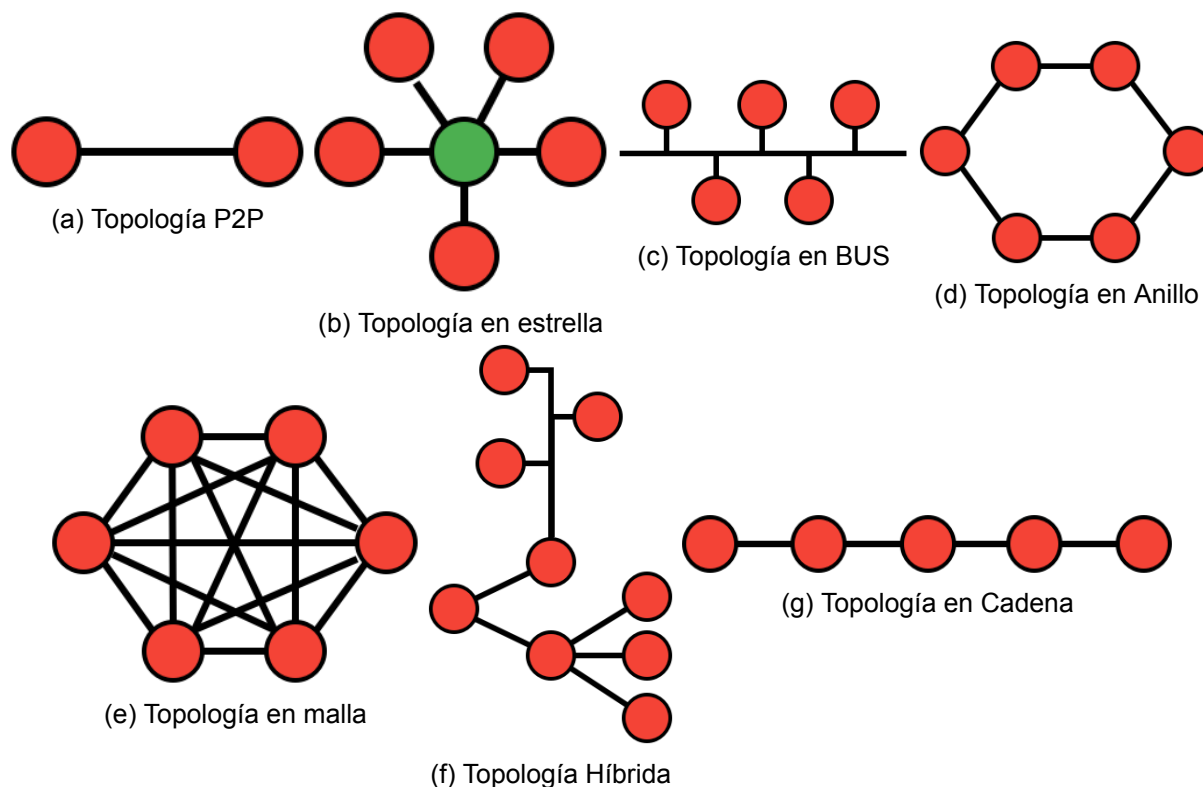


Figura 2.1: Distintas topologías de red

### 2.2.2. Nodos de una red

**Router o enrutador:** es un dispositivo de red que reenvía los paquetes mirando en la capa 3 del modelo OSI (IP) y conecta dos redes.

**Puente de red o bridge:** Funciona en la capa 2 del modelo OSI. Es un dispositivo que conecta dos segmentos de red formando una única subred, por lo que las dos “redes” pueden conectarse e intercambiar datos sin necesidad de un *router*.

**Conmutadores o switches:** dispositivo de red que filtra los datagramas del nivel 2 OSI (*Data Link Layer*, ver 2.1, pág. 6), también conocidos como *frames*, y reenvía los paquetes recibidos entre los puertos, dependiendo de la dirección MAC de cada *frame*. La diferencia entre un *switch* y un *hub* es que el *switch* sólo reenvía los paquetes por el puerto necesario. También existen un tipo especial de *switches* que pueden mirar en el nivel 3 OSI.

**Repetidores y hubs:** un repetidor es un dispositivo de red que, llegada una señal, limpia el ruido innecesario y la regenera. Un repetidor con múltiples puertos es un *hub*, trabajan en la capa 1 del modelo OSI (*Open Systems Interconnection* (Interconexión de Sistemas Abiertos)). Los repetidores requieren un pequeño tiempo para regenerar la señal, lo que puede crear un retardo en la señal.

**Interfaces de Red:** también conocido como tarjeta de red o *Network Interface Controller* (NIC), es un hardware, normalmente integrado en la placa base, que permite al ordenador conectarse a una red. Recibe el tráfico de una dirección de red. En las redes de Ethernet, tiene una dirección MAC (*Media Access Control* [Control de Acceso al Medio]) única. Estas direcciones son administradas por el IEEE (Instituto de Ingeniería Eléctrica y Electrónica)

evitando la duplicidad de estas. Cada dirección MAC ocupa 6 octetos, o 48 bits, a lo que suele ser representada como una cadena hexadecimal, por ejemplo: "43:31:50:30:74:33".

**Módem:** Dispositivos que transforman señales analógicas a digitales y viceversa. Son usados mayoritariamente en el ADSL (*Asymmetric Digital Subscriber Line* [Línea de Abonado Digital Asimétrica]).

**Cortafuegos o firewalls:** dispositivo que controla la seguridad mediante reglas de acceso. Aceptan determinados paquetes mientras rechazan otros. En una red doméstica, se puede poner un firewall que sólo acepte tráfico de los puertos de uso común (Páginas Web, e-mail, etc.) y rechace otros más peligrosos (Acceso remoto, SSH, SMTP, SOCKS...).

### 2.2.3. Enlaces de red

Según el modelo OSI, los enlaces de red corresponden a las capas 1 y 2. El medio físico puede ser tanto ondas de radio (Wi-Fi), como fibra óptica (FTTH) o impulsos de red (PLC, Ethernet, DSL).

#### Cableado

**Coaxial:** Cables de cobre o aluminio recubiertos de aislante, rodeado de un conductor, así se reducen las interferencias y la distorsión. Normalmente son usados para la transmisión de radio y TV, pero pueden ser usados para redes informáticas. Pueden llegar hasta a 500 Mbit/s <INSERTAR IMAGENES>

**Par trenzado o Ethernet:** Es el más usado en redes locales. Es un cable formado por finos cables trenzados en pares. En telefonía se usa el RJ11 o 6P4C (6 posiciones, 4 conectores) formado por 2 pares. Para ordenadores, según el estándar *Ethernet* se usa 8P8C o RJ45 de 4 pares, debido al nombre del estándar, este cable suele ser comúnmente llamado "cable de Ethernet". Puede llegar hasta 10 Gbit/s

**Fibra óptica:** Hilo de cristal o plástico flexible que permite que la luz se refleje en su interior, transmitiéndola de un extremo a otro del cable. No tienen apenas pérdida por distancia y son inmunes a las interferencias electromagnéticas. Además, permiten varias frecuencias de onda, lo que equivale a una transferencia de datos más rápida. Son usados para salvar las largas distancias entre continentes.

#### Comunicación inalámbrica o Wireless

**Microondas terrestres:** Transmisores, receptores y repetidores terrestres que operan en frecuencias de entre 300 MHz y 300 GHz de propagación de alcance visual, por lo que los repetidores no se separan más de 48 km.

**Comunicación satelital:** Microondas y ondas de radio que no sean reflejadas por la atmósfera terrestre. Los satélites mantienen una órbita geosíncrona, es decir, el periodo de rotación es el mismo que el de la tierra, lo que se produce a una altura de 35786 km.

**Celular o PCS:** Ondas electromagnéticas de entre 1800 y 1900 MHz. Son las usadas por los teléfonos móviles. A partir del 2G o GPRS, se podía acceder a Internet con de TCP/IP. El sistema divide la cobertura en áreas geográficas, cada una con un repetidor. Repiten los datos entre un repetidor y el otro.

**Ondas de radio:** Ondas de 0.9, 2.4, 3.6, o 5 GHz. El estándar más usado es el *IEEE 802.11*, también conocido como wifi o Wi-Fi que opera en la banda de 2.4 GHz, a excepción de la versión IEEE 802.11ac que opera a 5GHz que tiene menos interferencias, pero también menor alcance.

## 2.3. Paquetes de red

Es cada serie de bits en la que se divide la información enviada por una red. Según el modelo OSI, un paquete es estrictamente el PDU de la capa de red. El paquete de red se encuentra encapsulado en la capa anterior del modelo OSI. Por ejemplo, en estándares de comunicación TCP/IP, un segmento TCP puede ser llevado por varios paquetes IP transportados por varios frames de Ethernet. Está formado por varios protocolos y en él se distinguen tres partes:

**Header** o cabecera: Datos e información sobre el paquete. (Dirección IP, MAC, versión, etc)

**Payload** o carga: Los datos que se quieren transferir.

**Trailer** o cola: En ocasiones es inexistente (como en UDP) pero suele ser un código de comprobación de errores.

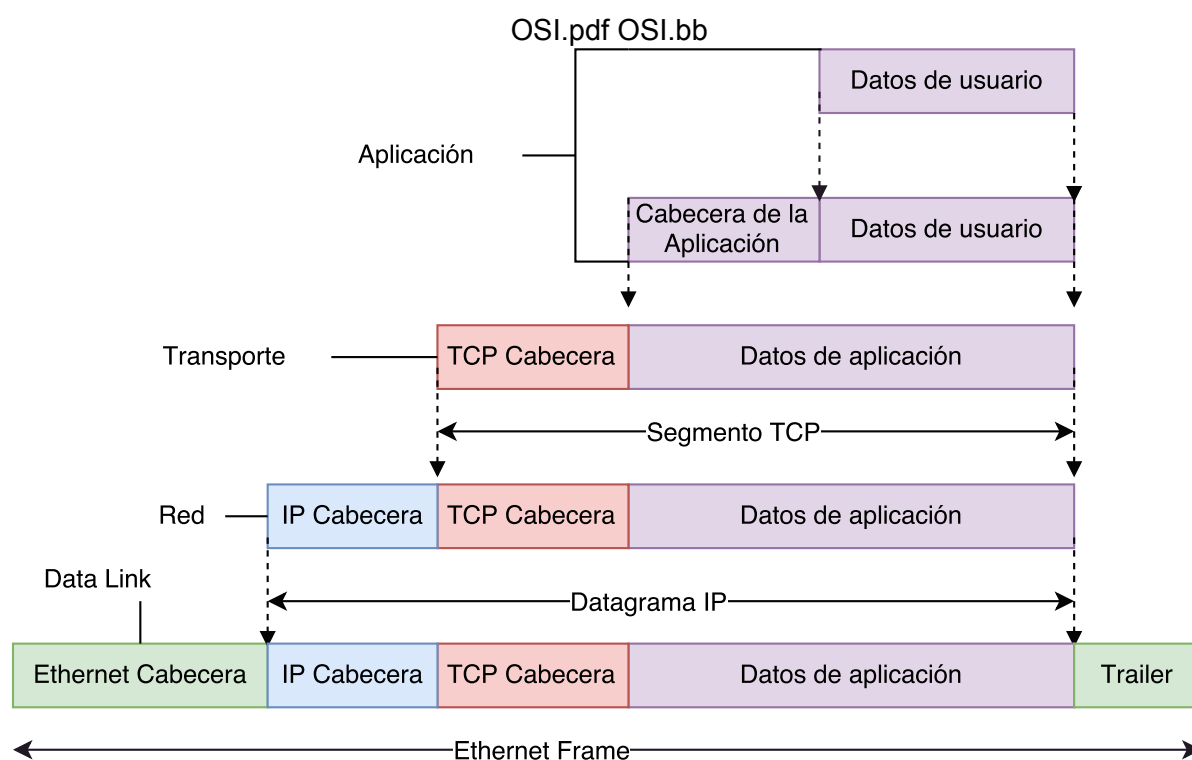


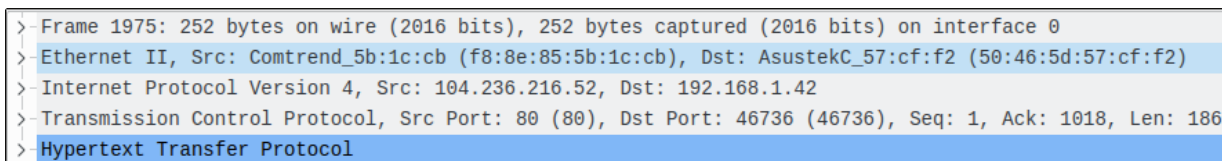
Figura 2.2: Encapsulación de red. El Datagrama IP es lo considerado 'Paquete de red'

### 2.3.1. Ejemplo: Paquete de red

## 2.4. Protocolos

Un protocolo de comunicación es un conjunto de reglas para intercambiar información entre enlaces de red. En una pila de protocolos, cada protocolo cubre los servicios del protocolo de la

capa anterior. Por ejemplo, un e-mail se envía mediante el protocolo POP3 (*Post Office Protocol*, Protocolo de Oficina Postal) en la capa de Aplicación, sobre TCP en la capa de transporte, sobre IP en la capa de Red, sobre Ethernet para la capa *Data Link*.



```

> Frame 1975: 252 bytes on wire (2016 bits), 252 bytes captured (2016 bits) on interface 0
> Ethernet II, Src: Comtrend_5b:1c:cb (f8:8e:85:5b:1c:cb), Dst: AsustekC_57:cf:f2 (50:46:5d:57:cf:f2)
> Internet Protocol Version 4, Src: 104.236.216.52, Dst: 192.168.1.42
> Transmission Control Protocol, Src Port: 80 (80), Dst Port: 46736 (46736), Seq: 1, Ack: 1018, Len: 186
> Hypertext Transfer Protocol
  
```

Figura 2.3: Captura de pantalla de Wireshark (Véase 1.1.7, pg. 5) en la que se muestran los protocolos que forman un paquete de red HTTP.

### 2.4.1. Familia de protocolos de internet

También conocido como *Internet Protocol Suite*, y más conocido como TCP/IP, es el fundamento de las redes informáticas. Se trata de un conjunto de más de 100 protocolos que permiten la conexión de ordenadores tanto en Internet como en LAN, incluyendo protocolos de las aplicaciones más usadas.

#### Aplicación

Es la capa en la que se envían los datos a otras aplicaciones en otro ordenador o en el mismo. Las aplicaciones hacen uso de las capas inferiores para asegurarse que los datos lleguen a su destino. Algunos de los protocolos más usados son:

- **HTTP Hypertext Transfer Protocol:** Protocolo de Transferencia de Hipertexto. Es el protocolo base de la World Wide Web. Se trata de texto estructurado que usa hiperenlaces entre nodos que también contienen texto. El cliente, al entrar en una URL (*Uniform Resource Identifier*, Identificador de Recursos Uniforme), el agente de usuario (navegador) envía al servidor una petición de la página web, mediante HTTP. El servidor, envía como respuesta un documento HTML u otro recurso.
- **DNS Domain Name System:** Sistema de Nombres de Dominio. Un servidor DNS almacena una base de datos distribuida y jerárquica con información sobre el nombre del dominio y la dirección IP a la que está vinculada. Al intentar conectar a `http://www.4chan.org`, el cliente pregunta al servidor cual es la dirección IP asociada a esa dirección, y se conecta a tal IP, en este caso 104.16.66.203. Para evitar tener que consultar continuamente con el servidor, se almacenan en una caché en el cliente.
- **TLS/SSL Transport Layer Security**, y su predecesor *Secure Sockets Layer*. <VER APARTADO DE SEGURIDAD>
- **HTTPS** HTTP Seguro. Es HTTP con TLS aplicado.
- **DHCP Dynamic Host Configuration Protocol:** Protocolo de configuración dinámica del host. Este protocolo es controlado por un servidor DHCP que envía parámetros de configuración automática a los clientes. El ejemplo más común es el de cualquier Router doméstico, que asigna automáticamente a cada dispositivo una dirección IP diferente, pero dejando un rango en el que se pueden establecer IP's estáticas.

- **FTP *File Transfer Protocol*:** Protocolo de Transferencia de Archivos, te permite enviar archivos entre un cliente y un servidor. El protocolo TLS aplicado a FTP se denomina FTPS. Te permite acceder, mediante un usuario y contraseña, o de forma anónima, a un sistema de archivos jerárquico con nombres de archivo codificados. Utiliza el puerto 21 de forma predeterminada.
- **SSH *Secure Shell*:** Terminal seguro. Es un protocolo de red criptográfico que permite a un cliente conectarse a un servidor y ejecutar comandos de terminal como un usuario (conociendo el usuario y contraseña). Además, permite la creación de túneles, lo que permite asegurar cualquier aplicación a través de SSH, y el acceso a puertos bloqueados por el cortafuegos en el cliente. La mayoría de servidores de SSH incluyen un servidor de SFTP, el protocolo FTP con SSH aplicado.
- **IMAP *Internet Message Access Protocol*:** Protocolo de acceso a mensajes de Internet. Usa una conexión TCP/IP para conectarse a un servidor de e-mail y ver el contenido de los mensajes, sin necesidad de descargarlos. A diferencia de POP, te permite usar una bandeja de entrada desde varios clientes.

DHCP, DNS, FTP, HTTP, IMAP, POP, TLS/SSL, SMTP, RIP, SSH, Telnet

## Transporte

- **TCP *Transmission Control Protocol*:** Protocolo de Control de Transmisión. Se aplica a los paquetes para administrarles un orden y un sistema de comprobación de errores. Con todas las funcionalidades, ocupa bastante espacio, lo que aumenta la latencia, aunque es más fiable para el envío de la mayoría de los datos.
- **UDP *User Datagram Protocol*:** Es un protocolo muy minimalista. A diferencia del TCP, no garantiza que los paquetes lleguen, o lleguen en orden, o protección ante duplicados. Reduce mucho la latencia ya que no usa *handshaking*. Por ello es usado por ejemplo para *streamings* de televisión o videollamadas.

## Red

- **IP *Internet Protocol*:** Protocolo de Internet. Envía datagramas o paquetes de red a través de redes. Tiene una función de enrutamiento que es la que permite la interconexión de redes, y la existencia de Internet. Es un protocolo que encapsula el paquete definiendo en el *header* (cabecera) las direcciones IP del servidor y el cliente, o remitente y destinatario. La versión usada actualmente es IPv4 desarrollado en 1981, pero poco a poco se va abriendo paso la versión IPv6. La mayor diferencia es que la versión cuatro cuenta con direcciones de 32 bits lo que permite tan sólo unas 4.3 millardos ( $2^{32}$ ) de direcciones, mientras que la versión 6 tiene direcciones de 128 bits, lo que permite más de 340 sextillones ( $2^{128}$ ) de direcciones.
- **ICMP *Internet Control Message Protocol*:** Es un protocolo que no es usado por aplicaciones de usuario (a excepción de herramientas de diagnóstico como ping o traceroute). Lo usan los dispositivos de red, como los routers, para enviar notificaciones o mensajes de error indicando que un servicio no está disponible.

## Link

- **ARP *Address Resolution Protocol*:** Protocolo de resolución de direcciones. Es un protocolo que convierte direcciones de la capa de Red a la capa de Enlace (dir. IP a dir. MAC).

ARP, MAC, ETHERNET

## 2.5. Seguridad de redes

La seguridad de redes consiste en el conjunto de acciones que toma el administrador de redes para prevenir y evitar acceso no autorizado, mal uso, o caída del servicio de red.

### 2.5.1. Tipos de ataques

Hay dos tipos de ataques de red. Son ataques pasivos cuando el intruso intercepta los datos que viajan por la red, y se considera activo cuando el atacante modifica el funcionamiento normal de la red. Aquí algunos ejemplos de los ataques más comunes:

- **Ataques pasivos**

- Sniffing o analizador de paquetes:** Mediante un software se muestran los datos de los paquetes de red enviados y recibidos por la red.
- Escáner de puertos:** Se envían numerosas peticiones al servidor por los servidores más comunes, así se comprueba que puertos están abiertos. Por ello es recomendable cambiar los puertos por defecto de los servidores importantes.
- Escáner IDLE:** Se realiza un escáner de puertos para saber que servicios están disponibles, pero a través de otro ordenador "zombie", y observando el comportamiento de éste.

- **Ataques activos**

- Ataque de Denegación de Servicio:** Se "desborda" el ancho de banda mediante el envío de muchas peticiones a un servidor, además de ser de un tamaño excesivo.
- Ataque DDoS:** *Distributed Denial of Service*, o un ataque de Denegación de Servicio distribuido. Varios ordenadores hacen un ataque DoS a un mismo servidor, algunas veces los ordenadores forman parte de una botnet, y en ocasiones ocurre sin querer (al haber demasiado tráfico de red).
- Phishing:** Con el objetivo de obtener información como nombres de usuario y contraseña o tarjetas de crédito, se crea una página de apariencia parecida a la página que trata de simular. Los usuarios más incautos no notarán el cambio e introducirán sus datos en esta página.
- SQL Injection:** Es una técnica de inserción de código. Al pedir un servidor SQL datos como "Nombre" o "Apellido", se introduce junto a estos código malicioso que el servidor puede ejecutar. Por ejemplo, `SELECT * FROM alumnos WHERE nombre = '<nombreintroducido>'` ;. `<nombreintroducido>` puede ser Pablo o Juan, pero si se introduce `x'` ; `DROP TABLE alumnos;` `SELECT * FROM asignaturas WHERE 't' = 't'`, el código que interpreta el servidor eliminaría la tabla alumnos por completo.
- Ataque Smurf:** Es una especie de ataque DDoS. Se envían paquetes ICMP (probablemente pings) a distintas máquinas, pero estos paquetes que se envían, el valor de la dirección IP del remitente es la dirección IP del objetivo al que se quiere atacar. Por lo que, las máquinas a las que se las ha enviado el mensaje ICMP responderán todas al objetivo, haciendo así un DDoS.
- DNS poisoning:** Se modifica la caché de DNS de un ordenador, redireccionando a una IP incorrecta, de esta manera se puede realizar un ataque de phishing sin que lo sepa

el usuario del ordenador. En el caso de hacerlo con las tablas de ARP, se denomina *ARP Poisoning*.

## 2.5.2. Contramedidas

### Encriptación

Se suele denominar también E2EE o *End-to-end encryption*, es decir, encriptación de punto a punto. Se suelen usar claves PGP (*Pretty Good Privacy*, Privacidad bastante buena) para cifrar correos electrónicos y otros archivos. Para HTTP lo más común es la encriptación TLS, aunque también se está utilizando actualmente para email. El servidor genera o contiene una clave o certificado, luego el cliente, debe recibir o tener esa clave para poder desencriptar el mensaje.

### Cortafuegos

Primero necesitamos definir lo que es un **puerto**. Un puerto es un punto final de comunicación en un Sistema Operativo. El puerto siempre está asociado a una dirección IP y a un tipo de protocolo. Así completa el origen o destino de un paquete de red. Se aplica en la capa de transporte del modelo OSI. El puerto es un número de 16 bits, por lo que será un número comprendido entre 0 y 65536. Multitud de puertos están ya reservados por diversos protocolos y programas, como el 80 para HTTP, 22 para SSH o 25 para SMTP.

Un cortafuegos es un software que supervisa el tráfico de entrada y salida de datos, basado en unas reglas. Si un paquete de red cumple esas reglas, es rechazado. Pueden bloquear un paquete destinado a un puerto, de un protocolo (Bloquear SSH de Internet, pero no local), de una IP específica, entre otros atributos. También pueden configurarse en modo negativo o whitelist, aceptando tan sólo los paquetes que cumplan las reglas. Por ejemplo, puedes especificar que no acepte tráfico en el puerto 23. Pero igualmente puedes especificar que sólo acepte tráfico en el puerto 23.

```
>-Frame 1940: 356 bytes on wire (2848 bits), 356 bytes captured (2848 bits) on interface 0
>-Ethernet II, Src: AsustekC_57:cf:f2 (50:46:5d:57:cf:f2), Dst: 192.168.1.1 (f8:8e:85:5b:1c:cb)
>-Internet Protocol Version 4, Src: 192.168.1.42 (192.168.1.42), Dst: mailsrv5.dondominio.com (31.214.176.6)
>-Transmission Control Protocol, Src Port: 55190 (55190), Dst Port: 25 (25), Seq: 102, Ack: 298, Len: 290
>-Simple Mail Transfer Protocol
  >-Internet Message Format
    >-From: No-Reply <"administracion@ddavo.me">, 1 item
    >-To: Yo mismo <"david@ddavo.me">, 1 item
    |Subject: Tu cuenta en http://sitiodeejemplo.gov.es ha sido creada
    >-Content-Type: text/plain; charset="utf-8"
    |Content-Transfer-Encoding: 8bit
    |MIME-Version: 1.0
    >-Line-based text data: text/plain
      |Usuario: Ejemplo\r\n
      |Contrase\303\261a: tucontrase\303\261a\r\n
```

Figura 2.4: Captura de pantalla de Wireshark (Véase 1.1.7, pg. 5) en la que se muestra un paquete SMTP (email enviado) sin ningún tipo de encriptación. Se puede acceder a este paquete desde cualquier nodo de la red.



```

Hypertext Transfer Protocol
> POST /foros/ucp.php?mode=login HTTP/1.1\r\n
  Host: herramientas.educa.madrid.org\r\n
  Connection: keep-alive\r\n
  Content-Length: 153\r\n
  Cache-Control: max-age=0\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
  Origin: http://herramientas.educa.madrid.org\r\n
  Upgrade-Insecure-Requests: 1\r\n
  User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36\r\n
  Content-Type: application/x-www-form-urlencoded\r\n
  Referer: http://herramientas.educa.madrid.org/foros/ucp.php?mode=login&sid=aefe98686186ac00798319aae1ab9be2\r\n
  Accept-Encoding: gzip, deflate\r\n
  Accept-Language: es,en-US;q=0.8,en;q=0.6\r\n
  Cookie: _ga=GA1.2.1274426646.1466681918; phpbb3_2lj1k_u=1; phpbb3_2lj1k_k=; phpbb3_2lj1k_sid=aefe98686186ac00798319aae1ab9be2; style_cookie=null\r\n
  \r\n
[Full request URI: http://herramientas.educa.madrid.org/foros/ucp.php?mode=login]
[HTTP request 1/1]
[Response in frame: 3440]

HTML Form URL Encoded: application/x-www-form-urlencoded
> Form item: "username" = "usuariodeprueba"
> Form item: "password" = "asdfaag"
> Form item: "redirect" = "../ucp.php?mode=login"
> Form item: "sid" = "aefe98686186ac00798319aae1ab9be2"
> Form item: "redirect" = "index.php"
> Form item: "login" = "Identificarse"
> Form item: "username" = "usuariodeprueba"
> Form item: "password" = "asdfaag"

```

Figura 2.5: Otro ejemplo de captura de paquetes. Esta vez de un formulario de HTTP en el que personas autorizadas podrían ver el usuario y la contraseña.

# Capítulo 3

## El simulador de redes

### 3.1. Instalación

#### 3.1.1. Ubuntu / Debian

Tan sólo se debe descargar el paquete del programa. Para ello usa apt-get:

```
Descargas $ sudo apt-get install invproy
```

En caso de no estar en los repositorios, hay que hacerlo manualmente:

```
Descargas $ wget <url>
Descargas $ sudo dpkg -i InvProy.deb
Descargas $ invproy
```

Para iniciar el programa también puedes usar la lista de programas.

#### 3.1.2. Arch Linux

Puedes encontrar el programa en el AUR <ENLACE>, pero si nunca has instalado nada desde el AUR, debes seguir el siguiente procedimiento.

```
~ $ sudo pacman -S base-devel #Lo necesitas para compilar el paquete
#Ahora elige el sitio donde descargaras el paquete. Aquí no se va a instalar.
~ $ cd Builds
Builds $ curl -O <url> #Lo descargamos
Builds $ tar -xvzf invproy.tar.gz
Builds $ cd invproy
Invproy $ makepkg -sri
```

Y ya lo tendrías instalado en tu ordenador.

#### 3.1.3. Ejecución manual / instalación portable

Lo primero que necesitará es descargar las dependencias. Esto depende del Sistema Operativo. En el caso de GNU/Linux, sólo es necesario descargar python3-gobject. Después, clonamos el repositorio de git. Ejemplo en Ubuntu:

```
~ $ sudo apt-get update && sudo apt-get upgrade
~ $ sudo apt-get install git python3-gobject
~ $ cd Descargas
Descargas $ git clone https://github.com/daviddavo/InvProy.git
```

Una vez ya tenemos el repositorio de git clonado:

```
Descargas $ cd InvProy
Descargas $ python3 Main.py
```

En el caso de querer usar el programa con una interfaz gráfica, vamos con nuestro explorador de archivos a la carpeta donde queramos descargarlo. Abrimos una terminal y descargamos el programa con `git clone https://github.com/daviddavo/InvProy.git`. Luego entramos en la carpeta y ejecutamos el archivo `Main.py`

### **3.2. Uso del programa**

# Glosario y acrónimos

**ADSL** *Asymmetric Digital Subscriber Line* [Línea de Abonado Digital Asimétrica]

**Bit** *Binary digit*, o *dígito binario*. Cada dígito del sistema de numeración binario

**Botnet** Grupo de ordenadores coordinados conectados a un maestro mediante un virus. Gracias a este virus se pueden realizar tareas masivas como el envío de SPAM o ataques DDoS

**Bug** Error en un programa informático.

**Caché** Almacenamiento temporal de datos con el objetivo de reducir el retardo, la carga de los servidores y el ancho de banda consumido

**Capas de abstracción** Método de ocultar detalles de implementación de un set de funcionalidades

**Conmutación de paquetes** Método para enviar datos por una red de computadoras. Se divide el paquete en dos partes, una con información de control que leen los nodos para enviar el paquete a su destino y los datos a enviar

**Datos** Secuencia binaria de unos y ceros que contiene información codificada

**Dependencia** De un programa, otro tipo de software necesario para que éste funcione

**FTTH** *Fiber To The Home* [Fibra hasta el hogar]  
**FTTx** *Fiber to the X*

**GNU** *GNU's Not Unix* (GNU no es Unix)

**Hardware** Conjunto de elementos físicos o materiales que constituyen un sistema informático.

**IEEE** Instituto de Ingeniería Eléctrica y Electrónica

**International Organization for Standardization** Organización Internacional de Normalización. Compuesta de varias organizaciones nacionales se encarga

de la creación de estándares internacionales desde 1947.

**ISO** *International Organization for Standardization*

**LAN** *Local Area Network* [Red de Área Local]

**Librería** En informática, una librería o biblioteca es un conjunto de recursos y funciones diseñadas para ser usadas por otros programas. Incluyen plantillas, funciones y clases, subrutinas, código escrito, variables predefinidas...

**Linux** is a generic term referring to the family of Unix-like computer operating systems that use the Linux kernel

**MAC** *Media Access Control* [Control de Acceso al Medio]

**OSI** *Open Systems Interconnection* (Interconexión de Sistemas Abiertos)

**POP3** *Post Office Protocol*, Protocolo de Oficina Postal

**Programación imperativa** Las órdenes del programa cambian el estado de este mismo. Por ejemplo, una variable no tiene por que ser declarada con antelación y su valor es modificable. Es la que usa el código máquina de los ordenadores

**Repositorio** Servidor donde se alojan ficheros o archivos para su descarga

**Test** Lorem ipsum dolor sit amet

**Topología** "Rama de las matemáticas que trata especialmente de la continuidad y de otros conceptos más generales originados de ella, como las propiedades de las figuras con independencia de su tamaño o forma." [3][Topología]

**Topología de red** Configuración espacial o física de la red. (Ver 2.2 pág.6)

**URL** *Uniform Resource Identifier*, Identificador de Recursos Uniforme

# Bibliografía

- [1] BICSI. *Network Design Basics for Cabling Professionals*. 2002.
- [2] Robert Braden. *RFC 1122*. 1989.
- [3] Real Academia Española. *Diccionario de la lengua española, ed. XXIII*. 2014.
- [4] FSF. *Filosofía del Proyecto GNU*. 2013. url: <https://www.gnu.org/philosophy/philosophy.html>.
- [5] Roy Trennman & Maurice Moss. *The Internet*. 2009. url: <https://www.youtube.com/watch?v=iDbyYGrswtg>.
- [6] PSF. *What is Python? Executive Summary*. 2016. url: <https://www.python.org/doc/essays/blurb/>.

# Índice de figuras

1.1. <i>Gitflow</i> o flujo de trabajo de Git . . . . .	3
2.1. Distintas topologías de red . . . . .	8
2.2. Encapsulación de red. El Datagrama IP es lo considerado 'Paquete de red' . . . .	10
2.3. Captura de pantalla de Wireshark . . . . .	11
2.4. Wireshark: SMTP sin encriptación . . . . .	14
2.5. Wireshark: HTTP Form sin encriptación . . . . .	15

## Apéndice A

# Unidades de transferencia de datos

Cantidad de datos transferidos por unidad de tiempo. La unidad de tiempo es el segundo y la cantidad de datos puede ser medida en *bits* (bitrate), caracteres/símbolos (*baudrate*) o bytes (8 bits), en ocasiones también se utilizan *nibbles* (4 bits). Para expresar esta velocidad, se suelen usar múltiplos, que pueden ser en base binaria o decimal.

Se usa la “b” para designar los bits, y “B” para los Bytes. Después, se usan los prefijos del sistema internacional cuando es en base decimal, y los prefijos del SI cambiando la segunda sílaba por “bi” (e.g: kilobit / kibibit, kbit/s / Kibit/s) cuando se trata de múltiplos binarios.

### Tabla de múltiplos

Unidad	Símbolo	Equivalencia
Kilobit/s	kbit/s o kb/s	1000 bit/s
Megabit/s	Mbit/s o Mb/s	$10^6$ bit/s o $10^3$ kbit/s
Gigabit/s	Gbit/s o Gb/s	$10^9$ bit/s o $10^3$ Mb/s
Terabit/s	Tbit/s o TB/s	$10^{12}$ bit/s o $10^3$ Gb/s
Kibibit/s	Kibit/s	$2^{10}$ bit/s o 1024 bit/s
Mebibit/s	Mibit/s	$2^{20}$ bit/s o 1024 Kibit/s
Gibibit/s	Gibit/s	$2^{30}$ bit/s o 1024 Mibit/s
Tebibit/s	Tibit/s	$2^{40}$ bit/s o 1024 Gibit/s
Byte/s	Byte/s	8 bit/s
Kilobyte/s	kB/s	1000 Byte/s o 8000 bits/s
Megabyte/s	MB/s	$10^6$ Byte/s o 1000 kB/s
Gigabyte/s	GB/s	$10^9$ Byte/s o 1000 MB/s
Terabyte/s	TB/s	$10^{12}$ Byte/s o 1000 GB/s
Kibibyte/s	KiB/s	1024 Byte/s
Mebibyte/s	MiB/s	$2^{20}$ Byte/s
Gibibyte/s	GiB/s	$2^{30}$ Byte/s
Tebibyte/s	TiB/s	$2^{40}$ Byte/s

# Apéndice B

## Código del programa

### B.1. Main.py

```
1  # -*- coding: utf-8 -*-
2  #!/usr/bin/env python3
3
4  '''
5      InvProy - Simulador de Redes / Proyecto de Investigación
6      https://github.com/daviddavo/InvProy
7      Copyright (C) 2016 David Davó Laviña david@ddavo.me http://ddavo.me
8
9      This program is free software: you can redistribute it and/or modify
10     it under the terms of the GNU General Public License as published by
11     the Free Software Foundation, either version 3 of the License, or
12     (at your option) any later version.
13
14     This program is distributed in the hope that it will be useful,
15     but WITHOUT ANY WARRANTY; without even the implied warranty of
16     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17     GNU General Public License for more details.
18
19     You should have received a copy of the GNU General Public License
20     along with this program. If not, see <http://www.gnu.org/licenses/>.
21
22     //////////////////////////////////
23
24     Este programa es código libre: Puedes redistribuirlo y/o modificarlo
25     bajo los términos de la licencia GNU General Public License tal y como
26     publicado por la Free Software Foundation, ya sea la versión 3 de layout
27     licencia o la más reciente.
28
29     Este programa es distribuido con la esperanza de que sea útil, pero
30     SIN NINGUNA GARANTÍA; sin siquiera la garantía implícita de COMERCIABILIDAD
31     o de la APTITUD DE LA MISMA PARA UN PROPÓSITO PARTICULAR. Ver la GNU General
32     Public License para más detalles.
33
34     Debes haber recibido una copia de la GNU General Public License con
35     este programa, si no es así, ver <http://www.gnu.org/licenses/>.
36 '''
37 from datetime import datetime
38 startTime = datetime.now()
39
40 import configparser, os, csv, sys, time, random, math
41 import xml.etree.ElementTree as xmltree
42 from ipaddress import ip_address
43 from random import choice
44
45 #Esto hace que el programa se pueda ejecutar fuera de la carpeta.
46 startcwd = os.getcwd()
47
48 try:
49     os.chdir(os.path.dirname(sys.argv[0]))
50 except:
51     pass
52
53 os.system("clear")
54 print("\033[91m#####\033[00m")
55
56 print("InvProy Copyright (C) 2016 David Davó Laviña\ndavid@ddavo.me <http://ddavo.me>\n\
57 This program comes with ABSOLUTELY NO WARRANTY; for details go to 'Ayuda > Acerca de'\n")
```



```

58 This is free software, and you are welcome to redistribute it\n\
59 under certain conditions\n")
60
61 try: #Intenta importar los modulos necesarios
62     #sys.path.append("Modules/")
63     import Modules.Test
64 except:
65     print("Error: No se han podido importar los modulos...")
66     sys.exit()
67
68 #Aqui importamos los modulos del programa que necesitamos...
69
70 from Modules.logmod import *
71 from Modules import save
72
73 def lprint(*objects, sep=" ", end="\n", file=sys.stdout, flush=False):
74     print(*objects, sep=sep, end=end, file=file, flush=flush)
75     thing=str()
76     for i in objects:
77         thing += str(i) + sep
78     writeonlog(thing)
79
80 lprint("Start loading time: " + time.strftime("%H:%M:%S"))
81
82 try:
83     #Importando las dependencias de la interfaz
84     import gi
85     gi.require_version('Gtk', '3.0')
86     from gi.repository import Gtk, GObject, Gdk, GdkPixbuf
87 except:
88     lprint("Por favor, instala PyGObject en tu ordenador. \n En ubuntu suele ser 'apt-get install python3-gi'\n En
89         ↪ Archlinux es 'pacman -S python-gobject'")
90     sys.exit()
91
92 try:
93     import cairo
94 except:
95     print("Necesitas tener instalado cairo")
96     print("Como es lógico, pon 'pacman -S python-cairo' en Archlinux")
97     sys.exit()
98
99 #Definiendo un par de cosillas necesarias
100
101 gtk = Gtk
102 config = configparser.RawConfigParser()
103 configdir = "Config.ini"
104 config.read(configdir)
105 allobjects = []
106
107 #Funcion que convierte un numero a una str con [digits] cifras
108 def digitsnumber(number, digits):
109     if len(str(number)) == digits:
110         return str(number)
111     elif len(str(number)) < digits:
112         return "0" * (digits - len(str(number))) + str(number)
113     else:
114         return "-1"
115
116 #Convierte hexadecimal a RGBA tal y como Gdk lo requiere
117 def hex_to_rgba(value):
118     value = value.lstrip('#')
119     if len(value) == 3:
120         value = ''.join([v*2 for v in list(value)])
121     (r1,g1,b1,a1)=tuple(int(value[i:i+2], 16) for i in range(0, 6, 2))+(1,)
122     (r1,g1,b1,a1)=(r1/255.00000,g1/255.00000,b1/255.00000,a1)
123
124     return (r1,g1,b1,a1)
125
126 print("#42FF37", hex_to_rgba("#42FF37"))

```

```

127 #Comprueba la integridad del pack de recursos
128 def checkres(recurdir):
129     files = ["Cable.png", "Router.png", "Switch.png", "Computer.png", "Hub.png"]
130     cnt = 0
131     ss = []
132     for i in files:
133         if os.path.isfile(recurdir + i):
134             cnt += 1
135         else:
136             ss.append(i)
137
138     if not (cnt == len(files)):
139         lprint("WARNING!!!!111!!!!111!")
140         lprint("Faltan archivos en resources/"+recurdir)
141         lprint(ss)
142         sys.exit()
143     else:
144         lprint("Estan todos los archivos")
145
146 checkres(config.get("DIRS", "respack"))
147
148 #Envia a la Statusbar informacion.
149 contador = 0
150 def push_elemento(texto):
151     global contador
152     varra1 = builder.get_object("barra1")
153     data = varra1.get_context_id("Ejemplocontextid")
154     testo = time.strftime("%H:%M:%S") + " | " + texto
155     contador = contador + 1
156     varra1.push(data, testo)
157     writeonlog(texto)
158
159 #Retorna un entero en formato de bin fixed
160 def bformat(num, fix):
161     if type(num) == int:
162         return str("{0:0" + str(fix) + "b}".format(num))
163     else:
164         return "ERROR"
165
166 gladefile = "Interface2.glade"
167
168 try:
169     builder = Gtk.Builder()
170     builder.add_from_file(gladefile)
171     writeonlog("Cargando interfaz")
172     lprint("Interfaz cargada\nCargados un total de " + str(len(builder.get_objects())) + " objetos")
173     xmlroot = xmltree.parse(gladefile).getroot()
174     lprint("Necesario Gtk+ " + xmlroot[0].attrib["version"]+".0", end="")
175     lprint(" | Usando Gtk+ "
176           ↪ " +str(Gtk.get_major_version())+"."+str(Gtk.get_minor_version())+"."+str(Gtk.get_micro_version()))
177 except Exception as e:
178     lprint("Error: No se ha podido cargar la interfaz.")
179     if "required" in str(e):
180         xmlroot = xmltree.parse(gladefile).getroot()
181         lprint("Necesario Gtk+ " + xmlroot[0].attrib["version"]+".0", end="\n")
182         lprint(">Estas usando "
183               ↪ " +str(Gtk.get_major_version())+"."+str(Gtk.get_minor_version())+"."+str(Gtk.get_micro_version()))
184     else:
185         lprint("Debug: ", e)
186         sys.exit()
187
188 #Intenta crear el archivo del log
189 createlogfile()
190
191 #CONFIGS
192 WRES, HRES = int(config.get("GRAPHICS", "WRES")), int(config.get("GRAPHICS", "HRES"))
193 resdir = config.get("DIRS", "respack")
194 lprint(resdir)

```

```

195
196 #CLASSES
197
198 allkeys = set()
199 cables = []
200 clickedobjects = set() #Creamos una cosa para meter los ultimos 10 objetos clickados. (EN DESUSO)
201 clicked = 0
202 bttnclicked = 0
203 areweputtingcable = 0
204
205 #Función a medias, esto añadirá un objeto a la cola de ultimos objetos clickados, por si luego queremos deshacerlo o
    ↪ algo.
206 def appendtclicked(objeto):
207     clickedobjects.insert(0, objeto)
208     try:
209         clickedobjects.remove(9)
210     except:
211         pass
212
213 class MainClase(Gtk.Window):
214     def __init__(self):
215         global resdir
216
217         self.ventana = builder.get_object("window1")
218         self.ventana.connect("key-press-event", self.on_key_press_event)
219         self.ventana.connect("key-release-event", self.on_key_release_event)
220         self.ventana.set_default_size(WRES, HRES)
221         self.ventana.set_keep_above(bool(config.getboolean("GRAPHICS", "window-set-keep-above")))
222
223         builder.get_object("Revealer1").set_reveal_child(bool(config.getboolean("GRAPHICS",
    ↪ "revealer-show-default")))
224
225         i = int(config.get('GRAPHICS', 'toolbutton-size'))
226
227         #Probablemente estas dos variables se puedan coger del builder de alguna manera, pero no se cómo.
228         start = 3
229         end = 8
230         jlist = ["Router.png", "Switch.png", "Cable.png", "Computer.png", "Hub.png"]
231         for j in range(start, end):
232             objtmp = builder.get_object("toolbutton" + str(j))
233             objtmp.connect("clicked", self.toolbutton_clicked)
234             objtmp.set_icon_widget(Gtk.Image.new_from_pixbuf(Gtk.Image.new_from_file(resdir +
    ↪ jlist[j-start]).get_pixbuf().scale_simple(i, i, GdkPixbuf.InterpType.BILINEAR)))
235             objtmp.set_tooltip_text(jlist[j - start].replace(".png", ""))
236
237         global configWindow
238         #configWindow = cfgWindow()
239
240         builder.get_object("imagemenuitem1").connect("activate", self.new)
241         builder.get_object("imagemenuitem9").connect("activate", self.showcfgwindow)
242         builder.get_object("imagemenuitem1").connect("activate", self.new)
243         builder.get_object("imagemenuitem3").connect("activate", self.save)
244         builder.get_object("imagemenuitem4").connect("activate", self.save)
245         builder.get_object("imagemenuitem2").connect("activate", self.load)
246         builder.get_object("imagemenuitem10").connect("activate", about().show)
247         builder.get_object("show_grid").connect("toggled", self.togglegrid)
248
249     ### EVENT HANDLERS###
250
251     handlers = {
252         "onDeleteWindow":      exiting,
253         "onExitPress":         exiting,
254         "onRestartPress":      restart,
255
256     }
257     builder.connect_signals(handlers)
258
259     builder.get_object("toolbutton1").connect("clicked", objlst.show)
260
261     self.ventana.show_all()

```

```

262
263 class ObjLst():
264     def __init__(self):
265         self.view = builder.get_object("objetos_treeview")
266         self.tree = Gtk.TreeStore(str,str)
267         renderer = Gtk.CellRendererText()
268         column = Gtk.TreeViewColumn("Objetos", renderer, text=0)
269         self.view.append_column(column)
270         column.set_sort_column_id(0)
271
272         renderer = Gtk.CellRendererText()
273         column = Gtk.TreeViewColumn("Valor", renderer, text=1)
274         column.set_sort_column_id(1)
275         self.view.append_column(column)
276         self.view.set_model(self.tree)
277         self.view.show_all()
278
279         self.revealer = builder.get_object("Revealer1")
280         print("Revealer:",self.revealer.get_reveal_child())
281         self.panpos = 100
282
283     def append(self, obj, otherdata=[]):
284         #SI OBJ YA ESTÁ, QUE AÑADA ATRIBUTOS A LA LISTA.
285         it1 = self.tree.append(None, row=[obj.name, obj.objecttype])
286         it2 = self.tree.append(it1, row=["MAC", str(obj.macdir)])
287         itc = self.tree.append(it1, row=["Conexiones", "{}/{}/".format(len(obj.connections),
288             ↪ obj.max_connections)])
289         for i in otherdata:
290             self.tree.append(it1, row=i)
291
292         obj.trdic = {"MAC":it2, "Connections":itc}
293
294         return it1
295
296     def update(self, obj, thing, val):
297         if thing in obj.trdic.keys():
298             self.tree.set_value(obj.trdic[thing], 1, val)
299         else:
300             it = self.tree.append(obj.trlst, row=[thing, val])
301             obj.trdic[thing] = it
302
303     def upcon(self, obj):
304         if not hasattr(obj, "trcondic"):
305             obj.trcondic = {}
306         #objlst.tree.append(self.trdic["Connections"], row=[self.name, self.objecttype])
307         self.tree.set_value(obj.trdic["Connections"], 1, "{}/{}/".format(len(obj.connections),
308             ↪ obj.max_connections))
309         for i in obj.connections:
310             print(i.__repr__(), obj.trcondic)
311             if i in obj.trcondic.keys():
312                 self.tree.set_value(obj.trcondic[i], 0, i.name)
313             else:
314                 r = self.tree.append(obj.trdic["Connections"], row=[i.name, ""])
315                 obj.trcondic[i] = r
316
317     def show(self, *args):
318         rev = self.revealer.get_reveal_child()
319         if rev:
320             self.panpos = builder.get_object("paned1").get_position()
321
322         builder.get_object("paned1").set_position(-1)
323         self.revealer.set_reveal_child(not self.revealer.get_reveal_child())
324
325         if not rev:
326             pass
327
328     def set_value(self,*args):
329         self.tree.set_value(*args)
330
331     def delete(self, obj):

```

```

330         self.tree.remove(obj.trlst)
331
332     def showcfgwindow(self, *args):
333         global configWindow
334         try:
335             configWindow.show()
336         except:
337             configWindow = cfgWindow()
338             configWindow.show()
339
340     #24/06 Eliminada startCable(), incluida en toolbutton_clicked
341
342     def togglegrid(self, *widget):
343         widget = widget[0]
344         global TheGrid
345         obj = TheGrid.backgr_lay
346         if widget.get_active() != True and obj.is_visible():
347             obj.hide()
348         else:
349             obj.show()
350
351     #Una función para gobernarlos a todos.
352     def toolbutton_clicked(self, objeto):
353         global clicked
354         global bttnclicked
355         global areweputtingcable
356         if areweputtingcable != 0:
357             areweputtingcable = 0
358             push_elemento("Cancelada acción de poner un cable")
359
360         if objeto.props.label == "toolbutton5":
361             lprint("Y ahora deberíamos poner un cable")
362             push_elemento("Ahora pulsa en dos objetos")
363             areweputtingcable = "True"
364
365         object_name = objeto.props.label
366         clicked = True
367         bttnclicked = object_name
368
369     #Al pulsar una tecla registrada por la ventana, hace todo esto.
370     def on_key_press_event(self, widget, event):
371         keyname = Gdk.keyval_name(event.keyval).upper() #El upper es por si está BLOQ MAYUS activado.
372         global allkeys #Esta es una lista que almacena todas las teclas que están siendo pulsadas
373         if config.getboolean("BOOLEANS", "print-key-pressed") == True:
374             lprint("Key %s (%d) pulsada" % (keyname, event.keyval))
375             lprint("Todas las teclas: ", allkeys)
376         if not keyname in allkeys:
377             allkeys.add(keyname)
378         if ("CONTROL_L" in allkeys) and ("Q" in allkeys):
379             exiting(1)
380         if ("CONTROL_L" in allkeys) and ("R" in allkeys):
381             restart()
382         if ("CONTROL_L" in allkeys) and ("U" in allkeys):
383             global allobjects
384             print("HARD UPDATE")
385             print(allobjects)
386             for obj in allobjects:
387                 obj.update()
388
389         if ("CONTROL_L" in allkeys) and ("S" in allkeys):
390             global allobjects
391             MainClase.save()
392         if ("CONTROL_L" in allkeys) and ("L" in allkeys):
393             MainClase.load()
394             allkeys.discard("CONTROL_L")
395             allkeys.discard("L")
396
397     #Para no tener que hacer click continuamente
398     if ("Q" in allkeys):
399         self.toolbutton_clicked(builder.get_object("toolbutton3"))

```

```

400         if "W" in allkeys:
401             self.toolbutton_clicked(builder.get_object("toolbutton4"))
402         if "E" in allkeys:
403             self.toolbutton_clicked(builder.get_object("toolbutton5"))
404         if "R" in allkeys:
405             self.toolbutton_clicked(builder.get_object("toolbutton6"))
406         if "T" in allkeys:
407             self.toolbutton_clicked(builder.get_object("toolbutton7"))
408         return keyname
409
410     #Al dejar de pulsar la tecla deshace lo anterior.
411     def on_key_release_event(self, widget, event):
412         keynameb = Gdk.keyval_name(event.keyval).upper()
413         if config.getboolean("BOOLEANS", "print-key-pressed") == True:
414             lprint("Key %s (%d) released" % (keynameb, event.keyval))
415         global allkeys
416         allkeys.discard(keynameb)
417
418     def drag_drop(widget, context, x, y, time):
419         push_elemento("Drag drop at " + str(x) + ", " + str(y) )
420
421     #Comprueba si el objeto tiene una ip asignada
422     def has_ip(self):
423         try:
424             if self.IP != None:
425                 return True
426             else:
427                 return False
428         except:
429             return False
430
431     def save(*args):
432         global cables
433         global allobjects
434         lscl = 0
435         try:
436             if args[1].get_label() == "gtk-save-as":
437                 print("Guardando como")
438                 lscl = 1
439         except:
440             pass
441         save.save(allobjects,cables, aslc=lscl)
442         push_elemento("Guardando...")
443     def load(*args):
444         global cables
445         global allobjects
446         save.load(allobjects,cables)
447         push_elemento("Cargando...")
448     def new(*args):
449         global allobjects
450         global cables
451         save.last = 0
452         while len(allobjects) > 0:
453             allobjects[0].delete(pr=0)
454         while len(cables) > 0:
455             cables[0].delete()
456
457     def new(*args):
458         global cables
459         global allobjects
460         while len(allobjects) > 0:
461             allobjects[0].delete(pr=0)
462
463     #Esta clase no es mas que un prompt que pide 'Si' o 'No'.
464     #La función run() retorna 1 cuando se clicka sí y 0 cuando se clicka no, así sirven como enteros y booleans.
465     class YesOrNoWindow(Gtk.Dialog):
466         def __init__(self, text, *args, Yest="Sí", Not="No"):
467
468             self.builder = Gtk.Builder()
469             self.builder.add_from_file(gladefile)

```

```

470
471     self.yesornowindow = self.builder.get_object("YesOrNoWindow")
472     self.labeldialog = self.builder.get_object("YoN_label")
473     self.nobutton = self.builder.get_object("YoN_No")
474     self.yesbutton = self.builder.get_object("YoN_Yes")
475
476     self.nobutton.connect("clicked", self.on_button_clicked)
477     self.yesbutton.connect("clicked", self.on_button_clicked)
478
479     self.labeldialog.set_text(text)
480     self.yesbutton.set_label(Yest)
481     self.nobutton.set_label(Not)
482
483     self = self.yesornowindow
484
485     def on_button_clicked(self, widget):
486         dialog = self
487
488     def run(self):
489         return self.yesornowindow.run()
490         self.yesornowindow.hide()
491
492     def destroy(self):
493         self.yesornowindow.destroy()
494
495 objetocable1 = None
496
497 #Esto es el Grid donde van las cosicas. A partir de aqui es donde esta lo divertido.
498 class Grid():
499     def __init__(self):
500         #16/06/16 MAINPORT PASA A SER VARIAS LAYERS
501         self.overlay = builder.get_object("overlay1")
502         self.mainport = Gtk.Layout.new()
503         self.cables_layer = Gtk.Layout.new()
504         self.backgr_layer = Gtk.Layout.new()
505         self.select_layer = Gtk.Layout.new() #Aparecer un fondo naranja en la cuadrícula cuando se selecciona un objeto
506         self.animat_layer = Gtk.Layout.new() #La capa de las animaciones de los cables
507         self.overlay.add_overlay(self.backgr_layer)
508         self.overlay.add_overlay(self.select_layer)
509         self.overlay.add_overlay(self.cables_layer)
510         self.overlay.add_overlay(self.animat_layer)
511         self.overlay.add_overlay(self.mainport)
512
513         self.viewport = builder.get_object("viewport1")
514         self.eventbox = builder.get_object("eventbox1")
515         self.eventbox.connect("button-press-event", self.clicked_on_grid)
516         #self.viewport.get_hadjustment().set_value(800)
517
518         self.wres = config.getint("GRAPHICS", "viewport-wres")
519         self.hres = config.getint("GRAPHICS", "viewport-hres")
520         self.sqres = config.getint("GRAPHICS", "viewport-sqres")
521         self.overlay.set_size_request(self.wres*self.sqres, self.hres*self.sqres)
522
523         #Modifica el color de fondo del viewport
524         clr = hex_to_rgba(config.get("GRAPHICS", "viewport-background-color"))
525         print("CLR:", clr)
526         self.viewport.override_background_color(Gtk.StateFlags.NORMAL, Gdk.RGBA(*clr))
527
528         #13/07/16 Ahora esto va por cairo, mejooor.
529         ### INICIO CAIRO
530
531         width, height, sq = self.wres*self.sqres, self.hres*self.sqres, self.sqres
532         surface = cairo.ImageSurface(cairo.FORMAT_ARGB32, width, height)
533         ctx = cairo.Context(surface)
534         ctx.close_path ()
535         ctx.set_source_rgba(0,0,0,1)
536         ctx.set_line_width(1)
537
538         for i in range(self.wres):
539             ctx.move_to(i*sq, 0)

```

```

540         ctx.line_to(i*sq, height)
541     for i in range(self.hres):
542         ctx.move_to(0, i*sq)
543         ctx.line_to(width, i*sq)
544
545
546     ctx.stroke()
547     self.image = Gtk.Image.new_from_surface(surface)
548     ### FINAL DE LO DE CAIRO
549
550     self.backgr_lay.put(self.image, 0, 0)
551
552     def subshow(widget):
553         #Para que no aparezca arriba a la izquierda:
554         scrolled = builder.get_object("scrolledwindow1")
555         scrolled.get_vadjustment().set_value(height/3)
556         scrolled.get_hadjustment().set_value(width/3)
557
558     if config.getboolean("GRAPHICS", "start-centered"):
559         builder.get_object("window1").connect("show", subshow)
560     self.overlay.show_all()
561     self.contadorback = 0
562
563     def moveto(self, image, x, y, *args, layout=None):
564         if x < self.wres and y < self.hres:
565             if layout == None:
566                 layout = self.mainport
567             elif str(layout.__class__.__name__) == "Layout":
568                 layout = layout
569             else:
570                 print("layout.__class__.__name__", layout.__class__.__name__)
571             if image in layout.get_children():
572                 layout.move(image, x*self.sqres, y*self.sqres)
573             else:
574                 layout.put(image, x*self.sqres, y*self.sqres)
575         else:
576             print("\033[31mError: Las coordenadas se salen del grid\033[00m")
577
578     def clicked_on_grid(self, widget, event, *args):
579         global clicked
580         global bttnclicked
581         global allobjects
582         global areweputtingcable
583         self.contadorback += 1
584
585         push_elemento("Clicked on grid @" + str(self.gridparser(event.x, self.wres)) + "," +
586             ↪ str(self.gridparser(event.y, self.hres)))
587
588     if self.searchforobject(self.gridparser(event.x, self.wres), self.gridparser(event.y, self.hres)) == False:
589         if clicked == 1:
590             push_elemento("Clicked: " + str(clicked) + " bttnclicked: " + str(bttnclicked))
591             if bttnclicked == "Router":
592                 Router(self.gridparser(event.x, self.wres), self.gridparser(event.y, self.hres))
593                 push_elemento("Creado objeto router")
594             elif bttnclicked == "toolbutton4":
595                 Switch(self.gridparser(event.x, self.wres), self.gridparser(event.y, self.hres))
596                 push_elemento("Creado objeto switch")
597             elif bttnclicked == "toolbutton6":
598                 Computador(self.gridparser(event.x, self.wres), self.gridparser(event.y, self.hres))
599                 push_elemento("Creado objeto Computador")
600             elif bttnclicked == "toolbutton7":
601                 Hub(self.gridparser(event.x, self.wres), self.gridparser(event.y, self.hres))
602                 push_elemento("Creado objeto Hub")
603
604         elif self.searchforobject(self.gridparser(event.x, self.wres), self.gridparser(event.y, self.hres)) != False:
605             push_elemento("Ahí ya hay un objeto, por favor selecciona otro sitio")
606         else:
607             lprint("pls rebisa l codigo")
608             clicked = 0
609             bttnclicked = 0

```



```

609
610     #Button: 1== Lclick, 2== Mclick
611     #Para comprobar si es doble o triple click: if event.type == gtk.gdk.BUTTON_PRESS, o gtk.gdk_2_BUTTON_PRESS
612     if event.button == 3:
613         rclick_Object = self.searchforobject(self.gridparser(event.x, self.wres), self.gridparser(event.y,
614             ↪ self.hres))
615         if rclick_Object != False:
616             rclick_Object.rclick(event)
617         else:
618             print("Agua")
619
620     if areweputtingcable != 0:
621         objeto = self.searchforobject(self.gridparser(event.x, self.wres), self.gridparser(event.y, self.hres))
622         if objeto == False:
623             push_elemento("Selecciona un objeto por favor")
624         elif objeto != False:
625             if len(objeto.connections) < objeto.max_connections:
626                 if areweputtingcable == "True":
627                     push_elemento("Ahora selecciona otro más")
628                     areweputtingcable = "Secondstep"
629                     global objetocable1
630                     objetocable1 = objeto
631                 elif areweputtingcable == "Secondstep":
632                     push_elemento("Poniendo cable")
633                     areweputtingcable = 0
634                     global objetocable1
635                     cable = Cable(objetocable1, objeto)
636                     objeto.connect(objetocable1, cable)
637                     objetocable1 = 0
638             else:
639                 push_elemento("Número máximo de conexiones alcanzado")
640
641     #Te pasa las cordenadas int que retorna Gtk a coordenadas del Grid, bastante sencillito. Tienes que llamarlo 2
642     ↪ veces, una por coordenada
643     def gridparser(self, coord, cuadrados, mode=0):
644         if mode == 0:
645             partcoord = coord / self.sqres
646             for i in range(cuadrados + 1):
647                 if partcoord < i:
648                     return i
649             else:
650                 pass
651         if mode == 1:
652             return coord * self.sqres
653
654     def resizetogrid(self, image):
655         #Image debe ser una imagen gtk del tipo gtk.Image
656         pixbuf = image.get_pixbuf()
657         pixbuf = pixbuf.scale_simple(self.sqres, self.sqres, GdkPixbuf.InterpType.BILINEAR)
658         image.set_from_pixbuf(pixbuf)
659
660     #Una función para encontrarlos,
661     def searchforobject(self, x, y):
662         global allobjects
663         localvar = False
664         for i in range(len(allobjects)):
665             if allobjects[i].x == x:
666                 if allobjects[i].y == y:
667                     localvar = True
668                     objeto = allobjects[i]
669                     break
670         if localvar == True:
671             return objeto
672         else:
673             return False
674
675     def __str__(self):
676         lprint("No se que es esto")

```

```

677 TheGrid = Grid()
678
679 #Clases de los distintos objetos. Para no escribir demasiado tenemos la clase ObjetoBase
680 #De la que heredaran las demas funciones
681 cnt_objects = 1
682 cnt_rows = 2
683 objlst = MainClase.ObjLst()
684
685 import uuid
686
687 class ObjetoBase():
688     allobjects = []
689     cnt = 0
690     #Una función para atraerlos a todos y atarlos en las tinieblas
691     def __init__(self, x, y, objtype, *args, name="Default", maxconnections=4, ip=None):
692         global cnt_objects
693         global cnt_rows
694         global allobjects
695         global glade_file
696
697         #IMPORTANTE: GENERAR UUID PARA CADA OBJETO
698         #La v4 crea un UUID de forma aleatoria
699         self.uuid = uuid.uuid4()
700         print("\033[96mUUID:\033[00m", self.uuid)
701
702         self.builder = Gtk.Builder()
703         self.builder.add_from_file(glade_file)
704         self.menuemergente = self.builder.get_object("grid_rclick")
705         self.builder.get_object("grid_rclick-disconnect_all").connect("activate", self.disconnect)
706         self.builder.get_object("grid_rclick-delete").connect("activate", self.delete)
707         self.builder.get_object("grid_rclick-debug").connect("activate", self.debug)
708
709         allobjects.append(self)
710
711         self.realx = x * TheGrid.sqres
712         self.realy = y * TheGrid.sqres
713         self.x = x - 1
714         self.y = y - 1
715         self.connections = []
716         self.cables = []
717         self.max_connections = maxconnections
718
719         #Algún día pasaré todos los algoritmos a algoritmos de búsqueda binaria
720         for f in os.listdir(resdir):
721             lprint(f, f.startswith(objtype))
722             if f.startswith(objtype) and ( f.endswith(".jpg") or f.endswith(".png") ):
723                 self.imgdir = resdir + f
724                 break
725
726         self.image = gtk.Image.new_from_file(self.imgdir)
727         self.resizetogrid(self.image)
728         if name == "Default" or name == None:
729             self.name = self.objtype + " " + str(self.__class__.cnt)
730         else:
731             self.name = name
732         cnt_objects += 1
733         self.__class__.cnt += 1
734
735         TheGrid.moveto(self.image, self.x, self.y)
736         self.image.show()
737
738         self.maddir = mac()
739
740         print("MAC:", self.maddir, int(self.maddir), bin(self.maddir))
741         if ip == None:
742             print("No ip definida")
743             self.ipstr = "None"
744
745         #Ahora vamos con lo de aparecer en la lista de la izquierda,
746         #aunque en realidad es un grid

```

```

747     lista = objlst
748     self.trlst = lista.append(self)
749     self.image.set_tooltip_text(self.name + " (" + str(len(self.connections)) + "/" + str(self.max_connections)
750     ↪ + ")\\n" + self.ipstr)
751
752     self.window_changethings = w_changethings(self)
753     self.builder.get_object("grid_rclick-name").connect("activate", self.window_changethings.show)
754
755     self.cnt = 0 #Se me olvido que hace esta cosa
756
757 def load(self):
758     global cnt_objects
759     global cnt_rows
760     global allobjects
761     self.builder = Gtk.Builder()
762     self.builder.add_from_file(glade_file)
763     self.builder.get_object("grid_rclick").connect("activate", self.disconnect)
764     self.builder.get_object("grid_rclick-delete").connect("activate", self.delete)
765     self.builder.get_object("grid_rclick-debug").connect("activate", self.debug)
766     self.connections = []
767     self.cables = []
768     cnt_objects += 1
769     self.__class__.cnt += 1
770     allobjects.append(self)
771     self.image = gtk.Image.new_from_file(self.imgdir)
772     self.resizetogrid(self.image)
773     TheGrid.moveto(self.image, self.x-1, self.y-1)
774     self.image.show()
775
776     self.trlst = objlst.append(self)
777
778     self.image.set_tooltip_text(self.name + " (" + str(len(self.connections)) + "/" + str(self.max_connections)
779     ↪ + ")\\n" + self.ipstr)
780     self.window_changethings = w_changethings(self)
781     self.builder.get_object("grid_rclick-name").connect("activate", self.window_changethings.show)
782
783 #Esta funcion retorna una str cuando se usa el objeto. En lugar de <0XXXXXXXX object>
784 def __str__(self):
785     return "<Tipo: " + self.objectype + " | Name: " + self.name + " | Pos: " + str(self.x) + ", " + str(self.y) +
786     ↪ ">"
787
788 def debug(self, *args):
789     print("DEBUG")
790     print("MAC:", self.macdir, int(self.macdir))
791
792 def rclick(self, event):
793     global rclick_Object
794     rclick_Object = self
795
796     print(self)
797     lprint("rclick en", self.x, self.y, self.objectype, "\\nConnections: ", end="")
798     lprint(self.connections)
799     self.rmenu = self.menuemergente
800     if self.objectype == "Computer" and len(self.compcn()) > 0:
801         self.builder.get_object("grid_rclick-sendpkg").show()
802     else:
803         self.builder.get_object("grid_rclick-sendpkg").hide()
804     if len(self.connections) > 0:
805         self.builder.get_object("grid_rclick-disconnect").show_all()
806     else:
807         self.builder.get_object("grid_rclick-disconnect").hide()
808     self.rmenu.popup(None, None, None, None, event.button, event.time)
809
810 def resizetogrid(self, image, *args):
811     #Ver resizetogrid en Grid (clase)
812     lprint(*args)
813     TheGrid.resizetogrid(image)
814
815 def clickado(self, widget, event):

```

```

814         lprint("Clickado en objeto " + str(self) + " @ " + str(self.x) + ", " + str(self.y))
815
816     #Esta función se encarga de comprobar a que ordenador(es) está conectado
817     #en total, pasando por routers, hubs y switches.
818
819     #Nota, hacer que compruebe que ordenadores tienen IP, y cuales no.
820     def compcon(self, *args):
821         passedyet = []
822         comps = []
823         reself = self
824
825     def subcompcon(notself, *args):
826         nonlocal passedyet
827         nonlocal reself
828         subcomps = []
829
830         iterc = notself.connections
831         #print(notself, "connections:", iterc)
832         #next(ite rc)
833
834         for con in iterc:
835             if con.uuid != reself.uuid and con.uuid not in [obj.uuid for obj in passedyet]:
836                 passedyet.append(con)
837                 #print(con)
838                 if con.objecttype == "Computer":
839                     subcomps.append(con)
840                 elif con.objecttype == "Switch" or con.objecttype == "Hub":
841                     subcomps.extend(subcompcon(con))
842                 else:
843                     print("Saltado", con)
844                     pass
845                 #passedyet.append(con)
846
847         #print("passedyet", passedyet)
848         return subcomps
849
850     comps.extend(subcompcon(self))
851
852     try:
853         #comps.remove(self)
854         pass
855     except:
856         pass
857
858     if args == 1 or "Gtk" in str(args):
859         print("Comps:", comps)
860         print("\nCompsname:", [x.name for x in comps])
861
862     return comps
863
864     #Comprueba si un objeto está conectado a otro.
865     def isconnected(self, objeto):
866         cons = compcon(self)
867         if objeto in cons:
868             return True
869         else:
870             return False
871
872     #TODO: Para no tener que actualizar todo, que compruebe el que cambió
873     #TODO: !! Hacer que modifique el menu_emergente (Hecho a medias xds)
874     #Nota !!: No puedes buscar un objeto en una lista, debes buscar sus atr.
875     def update(self):
876         print("\033[95m>>Updating\033[00m", self)
877         print(self.builder.get_object("grid_rclick-disconnect"))
878         self.image.set_tooltip_text(self.name + " (" + str(len(self.connections)) + "/" + str(self.max_connections)
879                                     + ")")
880         objlst.set_value(self.trlst, 0, self.name)
881
882         objlst.update(self, "MAC", str(self.maddir))
883         for child in self.builder.get_object("grid_rclick-disconnect").get_submenu().get_children():

```

```

883         if child.props.label.upper() != "TODOS":
884             if child.link.uuid not in [x.uuid for x in self.connections]:
885                 print("Object", child.link.__repr__(), "in connections", self.connections)
886                 child.hide()
887                 child.destroy()
888             else:
889                 print("Object", child.link.__repr__(), "in self.connections", self.connections)
890         pass
891
892     objlst.upcon(self)
893
894     print("\033[95m<<\033[00m")
895
896     def connect(self, objeto, cable):
897         tmp = Gtk.MenuItem.new_with_label(objeto.name)
898         self.builder.get_object("grid_rclick-disconnect").get_submenu().append(tmp)
899         tmp.show()
900         tmp.connect("activate", self.disconnect)
901         #link es un objeto vinculado al widget, luego es útil.
902         tmp.link = objeto
903         tmp2 = Gtk.MenuItem.new_with_label(objeto.name)
904
905         if self.__class__.__name__ != "Switch" and self.__class__.__name__ != "Hub":
906             tmp2.connect("activate", self.send_pck)
907             tmp2.show()
908             tmp2.link = objeto
909
910         tmp = Gtk.MenuItem.new_with_label(self.name)
911         objeto.builder.get_object("grid_rclick-disconnect").get_submenu().append(tmp)
912         tmp.show()
913         tmp.connect("activate", objeto.disconnect)
914         tmp.link = self
915         tmp2 = Gtk.MenuItem.new_with_label(self.name)
916
917         if objeto.__class__.__name__ != "Switch" and objeto.__class__.__name__ != "Hub":
918             tmp2.show()
919             tmp2.connect("activate", objeto.send_pck)
920             tmp2.link = self
921
922         self.connections.append(objeto)
923         self.cables.append(cable)
924         #objlst.tree.append(self.trdic["Connections"], row=[objeto.name, objeto.objectype])
925
926         objeto.connections.append(self)
927         objeto.cables.append(cable)
928         #objlst.tree.append(objeto.trdic["Connections"], row=[self.name, self.objectype])
929
930         self.update()
931         objeto.update()
932
933         if objeto.__class__.__name__ == "Switch":
934             print("Connecting {} to {}".format(objeto, self))
935             objeto.connectport(self)
936         if self.__class__.__name__ == "Switch":
937             print("Connecting {} to {}".format(objeto, self))
938             self.connectport(objeto)
939
940     def disconnect(self, widget, *args, de=None):
941         print("Cables:", self.cables)
942         #QUICKFIX
943         try:
944             if widget.props.label.upper() == "TODOS" and de == None:
945                 de = "All"
946             elif de == None:
947                 de = widget.link
948         except:
949             print("NO WIDGET AT DISCONNECT()")
950
951         if de == "All":
952             ###NO FUNCIONA DEL TODO BIEN, NO USAR###

```

```

953     #Bug, el ultimo cable no se borra
954     print("Ahora a desconectar de todos")
955     while len(self.connections) > 0:
956         self.disconnect(widget, de=self.connections[0])
957
958     else:
959         objlst.tree.remove(self.trcondic[de])
960         del self.trcondic[de]
961         objlst.tree.remove(de.trcondic[self])
962         del de.trcondic[self]
963
964         de.connections.remove(self)
965         self.connections.remove(de)
966
967         iterc = iter(self.builder.get_object("grid_rclick-disconnect").get_submenu().get_children())
968         next(ite rc)
969         print("\033[91mLinks\033[00m", [x.link for x in iterc])
970
971         if de in [x.link for x in iterc]:
972             print("\033[91mSelf in\033[00m", self)
973
974         for cable in self.cables:
975             if cable.fromobj == self or cable.toobj == self:
976                 cable.delete()
977                 break
978
979         de.update()
980
981         if self.__class__.__name__ == "Switch":
982             self.disconnectport(de)
983         elif de.__class__.__name__ == "Switch":
984             de.disconnectport(self)
985
986     self.update()
987
988 def delete(self, *widget, conf=1, pr=1):
989     if pr == 1:
990         yonW = YesOrNoWindow("¿Estás seguro de que quieres eliminar " + self.name + " definitivamente? El objeto
991             ↪ será imposible de recuperar y te hechará de menos.")
992         yonR = yonW.run()
993         yonW.destroy()
994     else:
995         yonR = 1
996     if yonR == 1:
997         self.disconnect(0, de="All")
998         objlst.delete(self)
999         self.image.destroy()
1000         global allobjects
1001         allobjects.remove(self)
1002     elif yonR == 0:
1003         print("Piénsatelo dos veces")
1004     else:
1005         raise
1006
1007 def packet_received(self, pck, *args, port=None):
1008     print("Hola, soy {} y he recibido un paquete, pero no sé que hacer con él".format(self.name))
1009     if config.getboolean("DEBUG", "packet-received"):
1010         print(">Pck:", pck)
1011         if pck.frame != None:
1012             print("\033[91m>>Atributos del paquete\033[00m")
1013             totalen = pck.lenght + 14*8
1014             wfr = bformat(pck.frame, (totalen+14)*8)
1015             print(">Wfr:", wfr)
1016             mac1 = "{0:011b}".format(pck.frame)[0:6*8]
1017             print(">Mac:", int(mac1, 2))
1018             readmac = str(hex(int(mac1, 2))).strip("0x")
1019             print(":".join([readmac[i * 2:i * 2 + 2] for i, bl in enumerate(readmac[:2])]).upper())
1020
1021             print("<<Fin de los atributos")

```

```

1022 class mac():
1023     def __init__(self, *macaddr, bits=48):
1024         print("macaddr:", *macaddr)
1025         if macaddr == None or True:
1026             tmp = self.genmac(bits=bits)
1027
1028             self.int = tmp[0]
1029             self.str = tmp[1]
1030             self.bin = ("{:0:0"+str(bits)+"b}").format(self.int)
1031
1032     def genmac(*self, bits=48, mode=None):
1033         #Por defecto se usa mac 48, o lo que es lo mismo, la de toa la vida
1034         #Nota, falta un comprobador de que la mac no se repita
1035         realmac = int("11" + str("{0:0"+ str(bits-2) + "b}").format(random.getrandbits(bits-2)),2)
1036         readmac = str(hex(realmac)).upper().replace("0X", "")
1037         readmac = ":".join([readmac[i * 2:i * 2 + 2] for i,bl in enumerate(readmac[:2])])
1038         if mode == 0:
1039             return realmac
1040         if mode == 1:
1041             return readmac
1042         else:
1043             return [realmac, readmac]
1044
1045     def __str__(self):
1046         readmac = str(hex(self.int)).upper().replace("0X", "")
1047         return ":".join([readmac[i * 2:i * 2 + 2] for i,bl in enumerate(readmac[:2])])
1048
1049     def __bytes__(self):
1050         return Object.__bytes__(self)
1051
1052     def __int__(self):
1053         return self.int
1054     def __index__(self):
1055         return self.int
1056     def list(self):
1057         return self.str.split(":")
1058
1059 npack = 0
1060
1061 class Router(ObjetoBase):
1062     cnt = 1
1063     def __init__(self, x, y, *args, name="Default"):
1064         global cnt_objects
1065         self.objecttype = "Router"
1066         push_elemento("Creado Objeto Router")
1067
1068         ObjetoBase.__init__(self, x, y, self.objecttype, name=name)
1069         self.x = x
1070         self.y = y
1071
1072     def __del__(self, *args):
1073         push_elemento("Eliminado objeto")
1074         del self
1075
1076 ### ESTO ERA NESTED DE SWITHC ###
1077
1078 class Port():
1079     def __init__(self, switch):
1080         self.id = switch.portid
1081         self.dic = switch.pdic
1082         self.all = switch.pall
1083         switch.portid += 1
1084         self.switch = switch
1085         self.connection = None
1086         self.all[self.id] = self
1087         self.dic[self.id] = self.connection
1088     def connect(self, connection):
1089         self.connection = connection
1090         self.dic[self.id] = self.connection
1091     def disconnect(self):

```

```

1092         self.connection = None
1093         self.dic[self.id] = self.connection
1094     def is_available(self):
1095         if self.connection == None:
1096             return True
1097         return False
1098
1099 class w_switch_table(Gtk.ApplicationWindow):
1100     def __init__(self, switch):
1101         self.link = switch
1102         builder = switch.builder
1103         builder.get_object("window_switch-table_button").connect("clicked", self.hide)
1104         builder.get_object("window_switch-table").connect("delete-event", self.hide)
1105         self.store = Gtk.ListStore(str,int,int,int)
1106
1107         self.view = builder.get_object("window_switch-table-TreeView")
1108         self.view.set_model(self.store)
1109         for i, column_title in enumerate(["MAC", "Puerto", "TTL (s)"]):
1110             renderer = Gtk.CellRendererText()
1111             column = Gtk.TreeViewColumn(column_title, renderer, text=i)
1112             column.set_sort_column_id(i)
1113             self.view.append_column(column)
1114         self.ticking = False
1115         builder.get_object("window_switch-table").set_keep_above(True)
1116
1117     def show(self, *a):
1118         self.ticking = True
1119         GObject.timeout_add(1001, self.tick)
1120         for row in self.store:
1121             row[2] = row[3] - time.time()
1122         self.link.builder.get_object("window_switch-table").show_all()
1123
1124     def hide(self, window, *event):
1125         self.link.builder.get_object("window_switch-table").hide()
1126         self.ticking = False
1127         return True
1128     def append(self, lst):
1129         lst.append(lst[2])
1130         for row in self.store:
1131             row[2] = row[3] - time.time()
1132         print(lst)
1133         row = self.store.append(lst)
1134         print(self.view.get_property("visible"))
1135         if self.view.get_property("visible") == True:
1136             self.ticking = True
1137             GObject.timeout_add(1001, self.tick)
1138
1139     def tick(self):
1140         for row in self.store:
1141             row[2] = row[3] - time.time()
1142             if row[2] <= 0:
1143                 try:
1144                     self.store.remove(row.iter)
1145                     self.link.table.remove(row)
1146                 except:
1147                     pass
1148             if len(self.store) == 0:
1149                 self.ticking = False
1150         return self.ticking
1151     def remove(self, lst):
1152         for row in self.store:
1153             if row == lst:
1154                 self.store.remove(row.iter)
1155                 self.link.table
1156                 break
1157         pass
1158
1159 class Switch(ObjetoBase):
1160     cnt = 1
1161     #El objeto puerto

```



```

1162
1163 def __init__(self, x, y, *args, name="Default", maxconnections=5):
1164     self.objecttype = "Switch"
1165     self.portid = 0
1166     self.pdic = {}
1167     self.pall = {}
1168
1169     push_elemento("Creado objeto Switch")
1170     self.imgdir = resdir + "Switch.*"
1171     ObjetoBase.__init__(self, x, y, self.objecttype, name=name, maxconnections=maxconnections)
1172     self.x = x
1173     self.y = y
1174     self.timeout = 20 #Segundos
1175
1176     for p in range(self.max_connections):
1177         Port(self)
1178     print(self.pall)
1179
1180     self.table = [
1181         #[MAC, port, expiration]
1182     ]
1183     self.wtable = w_switch_table(self)
1184     child = Gtk.MenuItem.new_with_label("Routing Table")
1185     self.builder.get_object("grid_rclick").append(child)
1186     child.connect("activate", self.wtable.show)
1187     child.show()
1188
1189     self.ch = child
1190
1191 def load(self):
1192     ObjetoBase.load(self)
1193     del self.wtable
1194     self.table = []
1195     self.wtable = w_switch_table(self)
1196
1197     del self.ch
1198     child = Gtk.MenuItem.new_with_label("Routing Table")
1199     self.builder.get_object("grid_rclick").append(child)
1200     child.connect("activate", self.wtable.show)
1201     child.show()
1202
1203     self.ch = child
1204
1205
1206 def connectport(self, objeto):
1207     for port in self.pall:
1208         if self.pall[port].is_available():
1209             self.pall[port].connect(objeto)
1210             break
1211     print(self.pdic)
1212
1213 def disconnectport(self, objeto):
1214     for p in self.pdic:
1215         print("i: {}, idx: {}".format(p, self.pdic[p]))
1216         if objeto == self.pdic[p]:
1217             self.pall[p].disconnect()
1218             break
1219     print(self.pdic)
1220
1221 def packet_received(self, pck, port=None):
1222     macd = "{0:012b}".format(pck.frame)[0:6*8]
1223     macs = "{0:012b}".format(pck.frame)[6*8+1:6*16+1]
1224
1225     #LO PRIMERO: AÑADIRLO A LA TABLA
1226     readmac = str(hex(int(macs,2))).upper().replace("0X", "")
1227     readmac = ":".join([readmac[i * 2:i * 2 + 2] for i,bl in enumerate(readmac[:2])])
1228
1229     for tab in self.table:
1230         if tab[2] <= time.time():
1231             print("Ha llegado tu hora")

```

```

1232         self.table.remove(tab)
1233         self.wtable.remove(tab)
1234         if tab[0] == int(macdir,2):
1235             print("TAB[0] == mcd")
1236             tab[2] = int(time.time()+self.timeout)
1237             for row in self.wtable.store:
1238                 print(row[0], tab[0])
1239                 if int(row[0].replace(":", ""),16) == tab[0]:
1240                     row[3] = int(time.time()+self.timeout)
1241             if int(macs,2) not in [x[0] for x in self.table]:
1242                 tmp = [int(macs,2), port, int(time.time()+self.timeout)]
1243                 self.table.append(tmp)
1244                 tmp = [readmac, port, int(time.time()+self.timeout)]
1245                 self.wtable.append(tmp)
1246
1247         #####
1248
1249         #ObjetoBase.packet_received(self, pck)
1250
1251         ttl = int(pck.str[64:72],2)
1252         ttlnew = "{0:08b}".format(ttl-1)
1253         pck.str = "".join(( pck.str[:64], ttlnew, pck.str[72:] ))
1254
1255         print("self.macdir",int(self.macdir), int("{0:012b}".format(pck.frame)[6*8+1:6*16+1],2))
1256         print("TTL:", int(pck.str[64:72],2), pck.str[64:72])
1257
1258         print("Soy {} y mi deber es entregar el paquete a {}".format(self.name,int(macdir,2)))
1259         print("El paquete llegó por el puerto {}".format(port))
1260         dic = {}
1261         for i in self.connections:
1262             dic[int(i.macdir)] = i
1263         print("Connections MAC's:", dic)
1264
1265         #Cambiamos los bits de macs
1266         #Si macd en conn, enviarle el paquete
1267         #Si existe una tabla de enrutamiento que contiene una ruta para macd, enviar por ahi
1268         #Si no, enviar al siguiente, y así
1269         print(">MAAC:",int(macdir,2), "DIIIC:")
1270         if int(macdir,2) in dic and ttl > 0:
1271             pck.animate(self, dic[int(macdir,2)])
1272
1273         elif int(macdir,2) in [x[0] for x in self.table] and ttl >= 0:
1274             for x in self.table:
1275                 if x[0] == int(macdir,2):
1276                     pck.animate(self, self.pdic[x[1]])
1277
1278         elif "Switch" in [x.objectype for x in self.connections] and ttl >= 0:
1279             print("Ahora lo enviamos al siguiente router")
1280             print(int(macdir,2), dic)
1281             tmp1st = self.connections[:] #Crea una nueva copia de la lista
1282             print(tmp1st)
1283             for i in tmp1st:
1284                 if int(macs,2) == int(i.macdir):
1285                     print("REMOVING", i)
1286                     tmp1st.remove(i)
1287             try:
1288                 tmp1st.remove(*[x for x in tmp1st if x.objectype == "Computer"])
1289             except TypeError:
1290                 pass
1291             print("Tmp1st:", tmp1st)
1292             obj = choice(tmp1st)
1293             print("Sending to:", obj)
1294             pck.animate(self, obj)
1295
1296         def debug(self, *args):
1297             print(self.pdic)
1298             print("MyMac:", self.macdir)
1299             row_format = "{:>20}" * 3
1300             print(row_format.format("MAC", "NXT", "EXP s"))
1301             for row in self.table:

```

```

1302         if row[1] == None:
1303             row[1] = "None"
1304         if int(row[2]-time.time()) <= 0:
1305             self.table.remove(row)
1306             print(row_format.format(row[0], row[1], int(row[2]-int(time.time()))))
1307
1308     #¿Tengo permisos de escritura?, no se si tendré permisos
1309     #Update: Si los tenia
1310     class Hub(ObjetoBase):
1311         cnt = 1
1312         def __init__(self, x, y, *args, name="Default", maxconnections=4, ip=None):
1313             self.objecttype = "Hub"
1314             push_elemento("Creado objeto Hub")
1315             self.imgdir = resdir + "Hub.*"
1316             ObjetoBase.__init__(self, x, y, self.objecttype, name=name)
1317             self.x = x
1318             self.y = y
1319
1320         def packet_received(self, pck, port=None):
1321             ttl = int(pck.str[64:72], 2)
1322             macs = "{0:012b}".format(pck.frame)[6*8+1:6*16+1]
1323             ttlnew = "{0:08b}".format(ttl-1)
1324             pck.str = "".join(( pck.str[:64], ttlnew, pck.str[72:] ))
1325             if ttl >= 0:
1326                 for obj in self.connections:
1327                     pck.animate(self, obj)
1328
1329     class Computador(ObjetoBase):
1330         cnt = 1
1331         def __init__(self, x, y, *args, name="Default", maxconnections=1, ip=None):
1332             self.objecttype = "Computer"
1333
1334             push_elemento("Creado objeto Computador")
1335             self.img = resdir + "Comp.*"
1336             ObjetoBase.__init__(self, x, y, self.objecttype, name=name)
1337             self.x = x
1338             self.y = y
1339             self.max_connections = maxconnections
1340             self.IP = None
1341
1342             self.pingwin = PingWin(self)
1343             self.builder.get_object("grid_rclick-sendpkg").connect("activate", self.pingwin.show)
1344
1345             self.update()
1346
1347         def load(self):
1348             ObjetoBase.load(self)
1349             self.pingwin = PingWin(self)
1350             self.builder.get_object("grid_rclick-sendpkg").connect("activate", self.pingwin.show)
1351
1352     class ip():
1353         def __init__(self, *args, ipstr="None"):
1354             self.str = ipstr
1355
1356         def __str__(self):
1357             return self.str
1358
1359         def set_str(self, str):
1360             self.str = str
1361             self.parser(str, 0)
1362
1363         def set_bin(self, binar):
1364             t = binar
1365             print(bin(t))
1366             if "0b" not in str(t) and "." in str(t):
1367                 print("Type is str")
1368                 self.bins = t
1369             elif "0b" in str(bin(t)) and "." not in str(bin(t)):
1370                 print("Type is binar")
1371                 self.bin = t

```

```

1372         else:
1373             print("Error:", t)
1374             self.parser(t, 1)
1375
1376         #ip2p stands 4 'ip to parse'
1377         def parser(self, ip2p, mode):
1378             #mode 0: str2b
1379             if mode == 0:
1380                 tplst = ip2p.split(".")
1381                 toreturn = []
1382                 for i in tplst:
1383                     i = int(i)
1384                     toreturn.append("{0:08b}".format(i))
1385                 self.bins = ".".join(toreturn)
1386                 self.bin = int(self.bins.replace(".", ""), base=2)
1387                 return self.bins
1388
1389             #mode 1: b2str
1390             elif mode == 1:
1391                 if "0b" not in str(ip2p):
1392                     self.bin = bin(int(ip2p.replace(".", ""), base=2))
1393                     self.str = ".".join([str(int(i, base=2)) for i in ip2p.split(".")])
1394                 elif "0b" in str(ip2p):
1395                     print("La ip", ip2p, "es bin")
1396                     tmp = str(ip2p).replace("0b", "")
1397                     n = 8
1398                     self.bins = ".".join([tmp[i * n:i * n+n] for i,blah in enumerate(tmp[::n])])
1399                     self.str = ".".join([str(int(tmp[i * n:i * n+n], base=2)) for i,blah in enumerate(tmp[::n])])
1400                 else:
1401                     raise
1402             else:
1403                 print("Debug:", mode)
1404                 raise NameError('No mode defined')
1405
1406         def update(self):
1407             ObjetoBase.update(self)
1408             self.image.set_tooltip_text(self.name + " (" + str(len(self.connections)) + "/" + str(self.max_connections)
1409                 + ")\n" + str(self.IP))
1410             submenu1 = self.builder.get_object("grid_rclick-sendpkg").get_submenu()
1411             print("Compcon: ", [x.name for x in self.compcon()])
1412
1413             if self.IP != None:
1414                 objlst.update(self, "IP", str(self.IP))
1415
1416         #Ahora es cuando viene la parte de haber estudiado.
1417         #SÓLO ENVÍA PINGS, (ICMP)
1418         sub_N = 0
1419         def send_pck(self, *widget, to=None):
1420             global npack
1421             Sub_N = Computador.sub_N
1422             #nonlocal sub_N
1423             de = self
1424             print(widget)
1425             if to == None:
1426                 to = widget[0].link
1427
1428             print("fnc send_pck from {} to {}".format(self.name, to.name))
1429
1430             if MainClase.has_ip(self) and MainClase.has_ip(to):
1431                 print("Continuando")
1432             else:
1433                 print("Un objeto no tiene IP")
1434                 yonW = YesOrNoWindow("Uno o los dos objetos no tienen dirección IP", Yest="OK", Not="Ok también")
1435                 yonR = yonW.run()
1436                 yonW.destroy()
1437                 raise Exception("Un objeto no tiene IP")
1438
1439             #Ambos deben tener direccion ip
1440             #def __init__(self, header, payload, trailer, cabel=None):
1441             ping = Ping.create(0, self.IP, to.IP)
1442             Sub_N += 1

```

```

1441         npack += 1
1442
1443         print("PCK ICMP HEADER:", "{0:064b}".format(ping.icmp_header))
1444         print("PCK IPHEADER:", "{0:0160b}".format(ping.ip_header))
1445
1446         print("MAC's:", self.macdir, to.macdir)
1447         frame = eth(int(to.macdir), int(self.macdir), ping)
1448         frame.applytopack(ping)
1449         print("Pck frame:", ping.frame)
1450
1451         ping.animate(self, self.connections[0])
1452
1453         msg = "{} >Enviado ping a {}".format(time.strftime("%H:%M:%S"), str(to.IP))
1454         self.pingwin.statusbar.push(self.pingwin.statusbar.get_context_id(msg), msg)
1455
1456         #Ver routing: https://en.wikipedia.org/wiki/IP_forwarding
1457         def packet_received(self, pck, *args, port=None):
1458             print("Hola, soy {} y he recibido un paquete, tal vez tenga que responder".format(self.name))
1459             #Si el tipo de ping es x, responder, si es y imprimir info
1460             if config.getboolean("DEBUG", "packet-received"):
1461                 print(">Pck:", pck)
1462                 if pck.frame != None:
1463                     frame="{0:0111b}".format(pck.frame)
1464                     print("\033[91m>>Atributos del paquete\033[00m")
1465                     totalen = pck.lenght + 14*8
1466                     print("Frame:", bin(pck.frame))
1467                     mac1 = "{0:0111b}".format(pck.frame)[0:6*8]
1468                     readmac = str(hex(int(mac1,2))).strip("0x")
1469                     print(">Mac1:", ":".join([readmac[i * 2:i * 2 + 2] for i,bl in enumerate(readmac[::2])]).upper())
1470                     readmac = str(hex(int( "{0:0111b}".format(pck.frame)[6*8+1:6*16+1] ,2))).strip("0x")
1471                     print(">Mac2:", ":".join([readmac[i * 2:i * 2 + 2] for i,bl in enumerate(readmac[::2])]).upper())
1472                     print("EtherType:", int(frame[12*8+1:8*14+1],2))
1473                     print("Resto==Bits:", int(frame[8*14+1::],2)==pck.bits)
1474                     print(pck.str)
1475
1476                     n, tmp = 8, pck.str[96:128]
1477                     print("IPs:", ":".join([str(int(tmp[i * n:i * n+n], base=2)) for i,blah in enumerate(tmp[::n])]))
1478                     tmp = pck.str[128:160]
1479                     print("IPd:", ":".join([str(int(tmp[i * n:i * n+n], base=2)) for i,blah in enumerate(tmp[::n])]))
1480
1481                     print("<<Fin de los atributos")
1482                 n,tmp = 8, pck.str[128:160]
1483                 tmp = pck.str[128:160]
1484                 print(int(tmp,2), int(self.IP))
1485                 if int(tmp,2) == int(self.IP):
1486                     ty = int("{0:064b}".format(pck.icmp_header)[:8],2)
1487                     if ty == 8:
1488                         print("El paquete era para mí, voy a responder un gracias :D")
1489                         ping = Ping.create(1, self.IP, int(pck.str[96:128],2))
1490                         frame = eth(int("{0:0112b}".format(pck.frame)[6*8+1:6*16+1],2), int(self.macdir), ping)
1491                         frame.applytopack(ping)
1492
1493                         ping.animate(self, self.connections[0])
1494                     elif ty == 0:
1495                         print("De nada")
1496                     else:
1497                         print("ty es:", ty)
1498
1499                 msg = "{} >Recibido ping de {}".format(time.strftime("%H:%M:%S"), ":".join([str(int(tmp[i * n:i * n+n],
1500                                     ↵ base=2)) for i,blah in enumerate(tmp[::n])]))
1501                 self.pingwin.statusbar.push(self.pingwin.statusbar.get_context_id(msg), msg)
1502
1503         class Servidor(Computador):
1504             def __init__(self, x, y, *args, name="Default", maxconnections=1, ip=None):
1505                 self.objectype = "Servidor"
1506
1507                 push_elemento("Creado objeto {}".format(self.objectype))
1508                 self.img = resdir + "Server.*"
1509                 ObjetoBase.__init__(self, x, y, self.objectype, name=name)
1510                 self.x = x

```

```

1510     self.y = y
1511     self.max_connections = maxconnections
1512     self.IP = self.ip()
1513
1514 #La clase para los objetos cable
1515 class Cable():
1516     def __init__(self, fromo, to, *color):
1517         lprint("Argumentos sobrantes: ", *color)
1518         self.objectype = "Wire"
1519         self.fromobj = fromo
1520         self.toobj = to
1521         self.fromx = TheGrid.gridparser(fromo.x, TheGrid.wres,1)
1522         self.fromy = TheGrid.gridparser(fromo.y, TheGrid.hres,1)
1523         self.tox = TheGrid.gridparser(to.x, TheGrid.wres,1)
1524         self.toy = TheGrid.gridparser(to.y, TheGrid.hres,1)
1525         self.w = max(abs(fromo.realx - to.realx),3)
1526         self.h = max(abs(fromo.realy - to.realy),3)
1527
1528         self.cair()
1529
1530         self.x, self.y = min(fromo.x, to.x)-0.5, min(fromo.y, to.y)-0.5
1531
1532         TheGrid.moveto(self.image, self.x, self.y, layout=TheGrid.cables_lay)
1533         lprint("Puesto cable en: ", self.x, "; ", self.y)
1534
1535         self.image.show()
1536
1537         global cables
1538         cables.append(self)
1539         lprint("Todos los cables: ", cables)
1540
1541     def load(self):
1542         global cables
1543         self.cair()
1544         self.image.show()
1545         cables.append(self)
1546
1547         self.fromobj.connect(self.toobj, self)
1548
1549     def cair(self):
1550         fromo = self.fromobj
1551         to = self.toobj
1552         width, height = max(abs(self.fromobj.realx - self.toobj.realx),3), max(abs(self.fromobj.realy -
            ↪ self.toobj.realy),3)
1553         surface = cairo.ImageSurface(cairo.FORMAT_ARGB32, width, height)
1554         ctx = cairo.Context(surface)
1555
1556         #ctx.scale(width, height)
1557
1558         ctx.close_path ()
1559
1560         if config.getboolean("DEBUG", "show-cable-rectangle"):
1561             ctx.set_source_rgba(0, 0, 1, 0.1) # Solid color
1562             ctx.rectangle(0,0,width,height)
1563             ctx.fill()
1564
1565
1566         ctx.set_line_width(1.5)
1567         ctx.set_source_rgb(1,0,0)
1568         if (fromo.x < to.x and fromo.y < to.y) or (fromo.x > to.x and fromo.y > to.y):
1569             ctx.move_to(0, 0)
1570             ctx.line_to(width, height)
1571         elif fromo.x == to.x:
1572             ctx.move_to(width/2, 0)
1573             ctx.line_to(width/2, height)
1574         elif fromo.y == to.y:
1575             ctx.move_to(0, height/2)
1576             ctx.line_to(width, height/2)
1577         else:
1578             ctx.move_to(0, height)

```

```

1579         ctx.line_to(width, 0)
1580
1581     ctx.stroke()
1582
1583     self.image = gtk.Image.new_from_surface(surface)
1584     self.x, self.y = min(fromo.x, to.x)-0.5, min(fromo.y, to.y)-0.5
1585
1586     TheGrid.moveto(self.image, self.x, self.y, layout=TheGrid.cables_lay)
1587
1588     def delete(self):
1589         global cables
1590         cables.remove(self)
1591
1592         self.fromobj.cables.remove(self)
1593         self.toobj.cables.remove(self)
1594
1595         self.image.hide()
1596         print("\033[96mCable\033[00m", self, "\033[96mdeleted\033[00m")
1597         del self
1598
1599     save.classes = [ObjetoBase, Switch, Hub, Computador, Servidor, Cable]
1600
1601     #De momento sólo soportará el protocolo IPv4
1602     class packet():
1603         def __init__(self, header, trailer, payload, cabel=None):
1604             lprint("Creado paquete de res")
1605             self.header = header
1606             self.payload = payload
1607             self.trailer = trailer
1608             #self.packet = header + payload + trailer
1609
1610         def new_from_total(self, bits):
1611             print("Length (bits):", int(bin(bits)[18:33],2)*8)
1612             print("Real length:", int(len(bin(bits))-2 ))
1613             self.bits = bits
1614             self.lenght = int(bin(bits)[18:33],2)
1615             self.str = str("{0:0"+str(int(int(bin(bits)[18:33],2)*8 ))+"b}").format(self.bits)
1616             print(self.str)
1617
1618         def send(self, de):
1619             ##SIN TERMINAR##
1620             ##FALTA AÑADIR TODO LO DEL FRAME##
1621             if de.objecttype == "Computador":
1622                 to = de.connections[1]
1623                 self.animate(de, to)
1624
1625             #Siendo t=fps/s, v=px/s, v default = 84
1626             def animate(self, start, end, fps=30, v=200, color=None, port=None):
1627                 if color == None:
1628                     if self.color != None:
1629                         color = self.color
1630                     else:
1631                         color = "#673AB7"
1632                 from math import sqrt, pi
1633                 #Long del cable
1634                 try:
1635                     cable = start.cables[[x.toobj for x in start.cables].index(end)]
1636                 except ValueError:
1637                     cable = start.cables[[x.fromobj for x in start.cables].index(end)]
1638                 w, h = cable.w + TheGrid.sqres, cable.h + TheGrid.sqres
1639                 x, y = cable.x*TheGrid.sqres-TheGrid.sqres/2, cable.y*TheGrid.sqres-TheGrid.sqres/2
1640                 xi, yi = (start.x-0.5)*TheGrid.sqres-x, (start.y-0.5)*TheGrid.sqres-y
1641                 xf, yf = end.x, end.y
1642                 r = sqrt(cable.w**2+cable.h**2) #Píxeles totales
1643                 t=r/v #Tiempo en segundos que durara la animacion
1644                 tf = int(fps*t) #Fotogramas totales
1645                 spf = 1/fps #Segundos por fotograma
1646
1647                 sq = 12
1648                 surface = cairo.ImageSurface(cairo.FORMAT_ARGB32, w, h)

```

```

1649     ctx = cairo.Context(surface)
1650     ctx.close_path()
1651     ctx.set_source_rgba(0,1,1,1)
1652     ctx.arc(-sq/2,-sq/2,sq/2,0,2*pi)
1653     ctx.fill()
1654     ctx.stroke()
1655     ctx.close_path()
1656
1657     image = gtk.Image.new_from_surface(surface)
1658     TheGrid.animat_layer.put(image,x,y)
1659     TheGrid.animat_layer.show_all()
1660
1661     #print("x: {}, y: {}, tf:{}, spf*m:{}, t: {}".format(x/TheGrid.sqres,y/TheGrid.sqres,tf,int(spf*1000), t))
1662     f = 0
1663     x,y = xi,yi
1664     sx,sy = (w-TheGrid.sqres)/tf,(h-TheGrid.sqres)/tf
1665     if start.x > end.x:
1666         sx = -sx
1667     if start.y > end.y:
1668         sy = -sy
1669
1670     def iteration():
1671         nonlocal f
1672         nonlocal x
1673         nonlocal y
1674         nonlocal ctx
1675         nonlocal surface
1676         nonlocal port
1677         if f <= tf:
1678             #Do things
1679             #print("Current f: {}; x,y: {}, {}".format(f, x,y))
1680             x += sx
1681             y += sy
1682
1683             del ctx
1684             surface = cairo.ImageSurface(cairo.FORMAT_ARGB32, w, h)
1685             ctx=cairo.Context(surface)
1686             ctx.set_source_rgba(*hex_to_rgba(color))
1687             ctx.arc(x,y,sq/2,0,2*pi)
1688             ctx.fill()
1689             image.set_from_surface(surface)
1690
1691             f += 1
1692             return True
1693         else:
1694             del ctx
1695             image.destroy()
1696             del surface
1697             #print("Paquete enviado a {}".format(end))
1698             if end.__class__.__name__ == "Switch":
1699                 for p in end.pall:
1700                     if end.pall[p].connection == start:
1701                         port = p
1702                         break
1703                 print("PORT:", port)
1704                 end.packet_received(self,port=port)
1705                 return False
1706             end.packet_received(self, port=port)
1707             return False
1708
1709     GObject.timeout_add(spf*1000, iteration)
1710
1711
1712     return True
1713
1714     def __str__(self):
1715         return "<" + str(packet) + ">"
1716
1717 # ETHERNET LAYER #
1718 #Usando DIX, más comun en IP

```



```

1719 #Al ser emulado no es necesario CRC Checksum
1720 #SIEMPRE 112 longitud (48*2+16)
1721 class eth(packet):
1722     #Se crea el header
1723     def __init__(self, destmac, sourcemac, *pack, EtherType=0x0800):
1724         def corrector(mac):
1725             if type(mac) == str:
1726                 mac2 = 0
1727                 for x in mac.split(":"):
1728                     mac2 = mac2 << 8 | int(x, 16)
1729                 return mac2
1730             elif type(mac) == int:
1731                 return mac
1732             else:
1733                 raise Exception("MAC ERROR")
1734
1735         destmac = corrector(destmac)
1736         sourcemac = corrector(sourcemac)
1737         print("Destmac", "{0:048b}".format(destmac))
1738
1739         self.machheader = (destmac << (6*8+1) | sourcemac) << 16 | EtherType
1740         print(int("{0:0111b}".format(self.machheader)[0:6*8],2))
1741
1742     #Se le añade la payload al frame
1743     def applytopack(self, pack):
1744         self.pack = pack
1745         print(">Mach:", bin(self.machheader).replace("0b", ""))
1746         print(">Pck:", pack)
1747         print(pack.lenght)
1748         ret = (self.machheader << pack.lenght*8) | pack.bits
1749         pack.frame = ret
1750         pack.framesrt = None
1751         print("pack.len: {}, bits len: {}".format(pack.lenght*8, len(bin(pack.bits).strip("0b"))))
1752         print(">Ret:", bin(ret).replace("0b", ""))
1753         print(int("{0:0111b}".format(self.machheader)[0:6*8],2))
1754         return ret
1755
1756     def __str__(self):
1757         return str( bin(self.machheader) )
1758
1759 #Internet Layer
1760 class icmp(packet):
1761     def __init__(self, ipheader, icmpheader, payload):
1762         print("Len:", int(bin(ipheader)[18:33],2)-28)
1763         self.bits = (ipheader << 8*8 | icmpheader) << ( (int(bin(ipheader)[18:33],2) -28) * 8) | payload #BITS 16a31
1764         packet.new_from_total(self, self.bits)
1765
1766     def __str__(self):
1767         return self.str
1768
1769
1770 ### Application layer ###
1771
1772 #Estos paquetes pueden ser Request o Reply.
1773 #El header es de 20 bytes, la payload es de 8 + datos opcionales, pero el estándar es 64 bits.
1774 #Tipo de mensaje es 8 para request y 0 para reply. El ICMP es siempre 0.
1775 class Ping(icmp):
1776     identifi = 0
1777     def __init__(self):
1778         pass
1779
1780     def create(r, sourceip, desti_ip, *n, payload=int( 4.3*10**19 ) << 6 | 42, \
1781         flags=0b010, ttl=32):
1782         self = Ping()
1783         if r == 0:
1784             Type = 8
1785             self.color = "#4CAF50"
1786         if r == 1:
1787             Type = 0

```

```

1788         self.color = "#F44336"
1789
1790     self.payload = payload
1791
1792     vihlto = 0b0100010100000000
1793     #20 Ipheader + 8 ICMPHeader + Payload
1794     lenght = int( 20 + 8 + ( int(math.log(payload, 2))+1)/8 ) #In Bytes
1795     frag_off = 0b00000000000000
1796     protocol = 1
1797     checksum = 0 #No es necesario porque no hay cables
1798     sourceip = int(sourceip)
1799     desti_ip = int(desti_ip)
1800     identific = Ping.identifi
1801     Ping.identifi += 1
1802
1803     self.ip_header = (((((((((vihlto << 16 | lenght)<<16 | identific) << 3 | flags) << 13 | frag_off) \
1804     << 8 | ttl) << 8 | protocol) << 16 | checksum) << 32 | sourceip) << 32 | desti_ip)
1805
1806     identifier = 1*2**15 + 42 * 2**8 + 42
1807     Code = 0
1808     icmp_header_checksum = random.getrandbits(16)
1809     self.icmp_header = (((((((Type << 8) | Code)<< 16) | checksum) << 16) | identifier) << 16) | identific)
1810     self.pck = icmp(self.ip_header, self.icmp_header, self.payload)
1811
1812     self.str = self.pck.str
1813     self.lenght = self.pck.lenght
1814     self.bits = self.pck.bits
1815
1816     return self
1817
1818
1819
1820 #Ventana para configurar las variables de Config.ini
1821 #Nota: Por terminar
1822 class cfgWindow(MainClase):#MainClase):
1823     def __init__(self, *args):
1824         push_elemento("Invocada ventana de configuracion")
1825         writeonlog("Has invocado a la GRAN VENTANA DE CONFIGURACION <--- Boss")
1826         self.cfgventana = builder.get_object("cfgwindow")
1827         self.cfgventana.connect("key-press-event", self.on_key_press_event)
1828         self.cfgventana.connect("key-release-event", self.on_key_release_event)
1829         self.cfgventana.connect("delete-event", self.hidewindow)
1830
1831         builder.get_object("button2").connect("clicked", self.save)
1832
1833         self.eraselogs = builder.get_object("eraselogs")
1834         self.eraselogs.connect("clicked", self.borrarlogs)
1835
1836         self.cfgbtttn1 = builder.get_object("checkbutton1")
1837         self.cfgbtttn1.connect("toggled", self.btntoggled)
1838         if config.getboolean("BOOLEANS", "print-key-pressed") == True:
1839             self.cfgbtttn1.set_active(True)
1840         else:
1841             self.cfgbtttn1.set_active(False)
1842
1843         booleans = {"print-key-pressed": "print-key-pressed"}
1844
1845         #TODO ESTO ES PARA LOS SPINNERS
1846
1847         #Todos los spinbuttons necesarios
1848         self.spinbuttons = [
1849             #[label, cfgsect, cfgkey, rangef, ranget, incrementf, increment],
1850             ["Win del wres", "GRAPHICS", "wres", 450, 1600, 5, 10],
1851             ["Win del hres", "GRAPHICS", "hres", 450, 1600, 5, 10],
1852             ["Wres del grid", "GRAPHICS", "viewport-wres", 20, 100, 1, 5],
1853             ["Hres del grid", "GRAPHICS", "viewport-hres", 15, 100, 1, 5],
1854             ["Res de los sq", "GRAPHICS", "viewport-sqres", 32, 128, 5, 10],
1855             ["Max logs", "DIRS", "Maxlogs", 3, 1000, 1, 5],
1856         ]
1857         self.createdspinbuttons = []

```

```

1858
1859     self.spinnergrid = builder.get_object("graph")
1860
1861     def forspin(spinner):
1862         spinbutton = Gtk.SpinButton.new(None, 0, 0)
1863         tplst = spinner
1864         label = Gtk.Label.new(tplst[0])
1865
1866         self.spinnergrid.insert_row(1)
1867
1868         #spinbutton.set_digits(0)
1869         spinbutton.set_numeric(True)
1870         spinbutton.set_range(tplst[3], tplst[4])
1871         spinbutton.set_increments(tplst[5], tplst[6])
1872         spinbutton.set_value(config.getfloat(tplst[1], tplst[2]))
1873
1874         #attach(child, left, top, width, height)
1875         self.spinnergrid.attach(label, 0, 1, 1, 1)
1876         self.spinnergrid.attach(spinbutton, 1, 1, 1, 1)
1877
1878         self.createdspinbuttons.append(spinbutton)
1879
1880     for spinner in self.spinbuttons:
1881         forspin(spinner)
1882
1883     #self.cfgventana.show_all()
1884
1885     def show(self, *args):
1886         self.cfgventana.show_all()
1887
1888     def on_key_press_event(self, widget, event):
1889         #global allkeys
1890         MainClase.on_key_press_event(self, widget, event)
1891         if "ESCAPE" in allkeys:
1892             push_elemento("Cerrada ventana de Configuracion")
1893             self.cfgventana.hide()
1894
1895         if ("CONTROL_L" in allkeys) and ("S" in allkeys):
1896             self.save()
1897             lprint(MainClase.on_key_press_event(self, widget, event))
1898
1899     def on_key_release_event(self, widget, event):
1900         MainClase.on_key_release_event(self, widget, event)
1901
1902     def btntoggled(self, *args):
1903         if self.cfgbtttn1.get_active() == True:
1904             push_elemento("print-key-pressed set True")
1905             config.set("BOOLEANS", "print-key-pressed", "True")
1906         if self.cfgbtttn1.get_active() == False:
1907             push_elemento("print-key-pressed set False")
1908             config.set("BOOLEANS", "print-key-pressed", "False")
1909
1910     def borrarlogs(self, *lala):
1911         #prompt = YesOrNoWindow("Seguro que quieres borrar los logs?")
1912         #if prompt.on_button_clicked(0) == True:
1913             push_elemento("Borrando logs")
1914             for the_file in os.listdir("logfiles/"):
1915                 file_path = os.path.join("logfiles/", the_file)
1916                 try:
1917                     if os.path.isfile(file_path):
1918                         os.unlink(file_path)
1919                 except e:
1920                     lprint(e)
1921
1922     def save(self, *args):
1923         #[label, cfgsect, cfgkey, rangef, ranget, incrementf, increment],
1924         lprint(self.createdspinbuttons)
1925         for i in range(len(self.createdspinbuttons)):
1926             tplst = self.spinbuttons[i]
1927             config.set(tplst[1], tplst[2], int(self.createdspinbuttons[i].get_value()))

```

```

1928
1929     push_elemento("Configuracion guardada")
1930     with open(configdir, 'w') as cfgfile:
1931         lprint("Guardando archivo de configuracion")
1932         try:
1933             config.write(cfgfile)
1934         except:
1935             lprint("Error al guardar la configuracion")
1936
1937     def hidewindow(self, window, *event):
1938         window.hide()
1939         return True
1940
1941 class w_changethings(): #Oie tú, pedazo de subnormal, que cada objeto debe tener una...
1942     #0 tal vez no sea necesario... A la hora de llamar a la función, espera ¿Con quien estoy hablando?
1943     #Nota, ver notas escritas en la mesa
1944     def __init__(self, objeto):
1945         self.window = objeto.builder.get_object("changethings")
1946         self.name_entry = objeto.builder.get_object("changethings_name-entry")
1947         self.imagebutton = objeto.builder.get_object("changethings_imagebutton")
1948         self.applybutton = objeto.builder.get_object("chg_apply")
1949         self.applybutton.connect("clicked", self.apply)
1950         self.cancelbutton = objeto.builder.get_object("chg_cancel")
1951         self.cancelbutton.connect("clicked", self.cancel)
1952         self.window.connect("delete-event", self.hidewindow)
1953         self.window.connect("key-press-event", self.on_key_press_event)
1954         self.window.connect("key-release-event", self.on_key_release_event)
1955         objeto.builder.get_object("chg_MAC-regen").connect("clicked", self.regenclicked)
1956         print(objeto.builder.get_object("chg_MAC-regen").set_image(gtk.Image.new_from_stock("gtk-refresh", 1)))
1957
1958         self.link = objeto
1959         self.image = Gtk.Image.new_from_pixbuf(objeto.image.get_pixbuf())
1960
1961     def filter_ip(entry):
1962         PingWin.filter_ip(0, entry)
1963
1964     def filter_numshex(widget):
1965         text = widget.get_text().strip()
1966         widget.set_text("".join([i for i in text if i in "0123456789ABCDEFabcdef"]))
1967
1968     objeto.builder.get_object("changethings_entry-IP").connect("changed", filter_ip)
1969
1970     for i in ["chg_MAC-entry" + str(x) for x in range(0,5)]:
1971         objeto.builder.get_object(i).connect("changed", filter_numshex)
1972
1973     if objeto.objecttype != "Computer":
1974         objeto.builder.get_object("changethings_box-IP").destroy()
1975         objeto.builder.get_object("grid_label-IP").destroy()
1976
1977     #self.applybutton.connect("clicked", self.apply)
1978     #self.cancelbutton.connect("clicked", self.cancel)
1979
1980     def show(self, *widget):
1981         print("widget:", self.link)
1982         self.window.show_all()
1983         self.imagebutton.set_image(self.image)
1984         self.name_entry.set_text(self.link.name)
1985         tplst = self.link.maddir.list()
1986         for i in tplst:
1987             tmpentry = self.link.builder.get_object("chg_MAC-entry" + str(tplst.index(i)))
1988             tmpentry.set_text(i)
1989
1990     #Hacer que muestre/oculte los campos de "IP"
1991     if self.link.objecttype == "Computer":
1992         try:
1993             self.link.builder.get_object("changethings_entry-IP").set_text(str(self.link.IP))
1994         except AttributeError: #Cuando no tiene una str definida
1995             raise
1996         pass
1997     except TypeError:

```

```

1998         raise
1999         pass
2000     except:
2001         raise
2002 else:
2003     pass
2004
2005 def apply(self, *npi):
2006     #acuerdate tambien de terminar esto
2007     #Nota: Hacer que compruebe nombres de una banlist, por ejemplo "TODOS"
2008     yonR = None
2009     lprint(npi)
2010
2011     self.link.name = self.name_entry.get_text()
2012     lprint([ self.link.builder.get_object(y).get_text() for y in ["chg_MAC-entry" + str(x) for x in range(0,6)]
2013             ↪ ])
2014     self.link.maddir.str = ":".join( [ self.link.builder.get_object(y).get_text() for y in ["chg_MAC-entry" +
2015             ↪ str(x) for x in range(6)] ])
2016     self.link.maddir.int = int(self.link.maddir.str.replace(":", ""), 16)
2017     self.link.maddir.bin = "{0:048b}".format(self.link.maddir.int)
2018     if self.link.objecttype == "Computer":
2019         try:
2020             iptemp = self.link.builder.get_object("changethings_entry-IP").get_text()
2021             if iptemp == "":
2022                 pass
2023             elif self.link.builder.get_object("changethings_entry-IP").tmp == 2:
2024                 self.link.IP = ip_address(iptemp)
2025             else:
2026                 yonW = YesOrNoWindow("{} no es una IP válida, por favor, introduzca una IP
2027                 ↪ válida".format(iptemp), Yest="OK", Not="Ok también")
2028                 yonR = yonW.run()
2029                 yonW.destroy()
2030         except:
2031             print(Exception)
2032             raise
2033
2034     lprint("self.link.name", self.link.name)
2035
2036     #self.link.image.set_tooltip_text(self.link.name + " (" + str(self.link.connections) + "/" +
2037     ↪ str(self.link.max_connections) + ")")
2038     self.link.update()
2039     self.window.hide()
2040     if yonR!=None:
2041         self.show()
2042
2043 def cancel(self, *npi):
2044     lprint(npi)
2045     self.window.hide()
2046
2047 def hidewindow(self, window, *event):
2048     window.hide()
2049     return True
2050
2051 def on_key_press_event(self, widget, event):
2052     #global allkeys
2053     MainClase.on_key_press_event(self, widget, event)
2054     if "ESCAPE" in allkeys:
2055         push_elemento("Cerrada ventana de Configuracion")
2056         self.window.hide()
2057
2058 def on_key_release_event(self, widget, event):
2059     MainClase.on_key_release_event(self, widget, event)
2060
2061 def regenclicked(self, widget):
2062     t = ObjetoBase.mac.genmac()[1].split(":")
2063     for i in t:
2064         tmpentry = self.link.builder.get_object("chg_MAC-entry" + str(t.index(i)))
2065         tmpentry.set_text(i)
2066         tmpentry.show()

```

```

2064 class PingWin(Gtk.ApplicationWindow):
2065     def __init__(self, obj):
2066         self.link = obj
2067         builder = obj.builder
2068         self.win = builder.get_object("PingWin")
2069         self.statusbar = builder.get_object("PingWin_Statusbar")
2070         self.entry = builder.get_object("PingWin_entry")
2071         self.entry.set_placeholder_text("192.168.1.XXX")
2072         self.ping = builder.get_object("PingWin_Button")
2073
2074         self.ping.connect("clicked", self.do_ping)
2075
2076         self.entry.connect("changed", self.filter_ip)
2077         self.win.connect("delete-event", self.destroy)
2078
2079     def filter_ip(self, entry):
2080         if entry.get_text().strip("") == "":
2081             entry.override_background_color(Gtk.StateFlags.NORMAL, Gdk.RGBA(*hex_to_rgba("#E57373")))
2082         else:
2083             entry.tmp = 0
2084             text = entry.get_text().strip()
2085             entry.set_text("".join([i for i in text if i in "0123456789."]))
2086             if max([len(x) for x in entry.get_text().split(".")]) > 3:
2087                 print("IP NO VÁLIDA")
2088                 entry.tmp = 1
2089             try:
2090                 if max([int(x) for x in entry.get_text().split(".") if x != ""]) > 254:
2091                     print("IP NO VÁLIDA")
2092                     entry.tmp = 1
2093             except ValueError:
2094                 pass
2095             except:
2096                 raise
2097             if len([x for x in entry.get_text().split(".") if x != ""]) == 4 and entry.tmp==0:
2098                 print("IP ACABADA")
2099                 entry.tmp = 2
2100                 entry.override_background_color(Gtk.StateFlags.NORMAL, Gdk.RGBA(*hex_to_rgba("#9CCC65")))
2101
2102             if entry.tmp == 1:
2103                 entry.override_background_color(Gtk.StateFlags.NORMAL, Gdk.RGBA(*hex_to_rgba("#E57373")))
2104             elif entry.tmp == 0:
2105                 entry.override_background_color(Gtk.StateFlags.NORMAL, Gdk.RGBA(*hex_to_rgba("#FFA726")))
2106
2107     def do_ping(self, widget):
2108         ip = self.entry.get_text()
2109         if self.entry.tmp == 2:
2110             print(self.link.compcon())
2111             to = None
2112             for x in self.link.compcon():
2113                 if ip == str(x.IP):
2114                     to = x
2115                     print("IP: {} from {} in compcon {}".format(ip, to, self.link.compcon()))
2116                     Computador.send_pck(self.link, to=to)
2117                     break
2118             if to == None:
2119                 yonW = YesOrNoWindow("La IP {} no se ha encontrado".format(ip), Yest="OK", Not="Ok también")
2120                 yonR = yonW.run()
2121                 yonW.destroy()
2122
2123         else:
2124             yonW = YesOrNoWindow("{} no es una IP válida, por favor, introduzca una IP válida".format(ip), Yest="OK",
2125                                     ↪ Not="Ok también")
2126             yonR = yonW.run()
2127             yonW.destroy()
2128
2129     def show(self, widget):
2130         self.win.show()
2131     def destroy(self, window, event):
2132         window.hide()
2133         return True

```

```

2133
2134 class about(Gtk.AboutDialog):
2135     def __init__(self):
2136         self.win = builder.get_object("AboutWindow")
2137         self.win.connect("delete-event", self.destroy)
2138         self.win.connect("response", self.destroy)
2139         self.win.add_credit_section("Tutores", ["Julio Sánchez"])
2140         #self.win.add_credit_section("Contribuidores", [""])
2141         self = self.win
2142     def show(self, *args):
2143         print("Showing")
2144         self.win.show()
2145     def destroy(self, *args):
2146         self.win.hide()
2147         return True
2148
2149
2150 #Esta clase te permitirá deshacer acciones, algún día de un futuro lejano.
2151 class Undo():
2152     def __init__(self):
2153         self.lastactions = []
2154
2155 #Esta la pongo fuera porque lo mismo la necesito en otra clase
2156
2157 def exiting(self, *ahfjah):
2158     global log
2159     savelog()
2160     lprint("End time: " + time.strftime("%H:%M:%S"))
2161     print ("Window closed, exiting program")
2162     Gtk.main_quit()
2163
2164 def restart(*args):
2165     global log
2166     savelog()
2167     lprint("End time: " + time.strftime("%H:%M:%S"))
2168     lprint("Restarting program")
2169     print("\033[92m#####\033[00m")
2170     os.chdir(startcwd)
2171     os.execl(sys.executable, sys.executable, *sys.argv)
2172
2173 def leppard():
2174     lprint("Gunter lieben glauben globen")
2175
2176 writeonlog("Esto ha llegado al final del código al parecer sin errores")
2177 writeonlog("0 no")
2178 MainClase()
2179
2180 lprint("Actual time: " + time.strftime("%H:%M:%S"))
2181 lprint("Complete load time: " + str(datetime.now() - startTime))
2182 push_elemento("Parece que esta cosa ha arrancado en tan solo " + str(datetime.now() - startTime))
2183 Gtk.main()
2184
2185 print("\033[92m#####\033[00m")

```

## B.2. Modules/logmod.py

```

1 #Tenia ganas de probar como va en Python esto de los modulos
2 import time, configparser, os
3 config = configparser.RawConfigParser()
4 configdir = "Config.ini"
5 config.read(configdir)
6
7 log = []
8 def writeonlog(thingtowrite, *otherthingstowrite):
9     global log
10     thingtowrite = time.strftime("%H:%M:%S") + "@" + thingtowrite
11     try:
12         thingtowrite += " | " + str(otherthingstowrite)

```

```

13     except:
14         pass
15     log.append(thingtowrite + "\n")
16     #if len(log) > 15:
17     #    savelog()
18
19
20 def savelog():
21     global log
22     with open(config.get("DIRS", "Log"), "a") as logfile:
23         logfile.writelines(log)
24         log = []
25
26 def createlogfile():
27     if not os.path.exists("logfiles/"):
28         os.makedirs("logfiles/")
29     nlogfiles = int(len(os.listdir("logfiles/")))
30     if nlogfiles >= int(config.get("DIRS", "Maxlogs")):
31         #print(nlogfiles)
32         #print(int(config.get("DIRS", "Maxlogs")) - nlogfiles)
33         #for i in range(abs(nlogfiles - int(config.get("DIRS", "Maxlogs")))):
34         while nlogfiles > int(config.get("DIRS", "Maxlogs")):
35             #Aqui pones que borre el archivo mas viejo
36             nlogfiles -= 1
37             log.append("Borrado: " + str(min(os.listdir("logfiles/")))+ "\n")
38             try:
39                 os.remove("logfiles/" + min(os.listdir("logfiles/")))
40             except OSError:
41                 print("\033[31mError de I/O en {}, borrar la carpeta de
42                     ↪ logfiles\033[00m".format(str(OSError.filename)))
43             except:
44                 raise
45         try:
46             newlogfile = "logfiles/" + time.strftime("%y%m%d%H%M%S") + " " + config.get("DIRS", "Log")
47             try:
48                 os.rename("Log.log", newlogfile)
49             except:
50                 print('!Ojo cuidao que no se ha podido renombrar <Log.log>')
51         except:
52             pass

```

### B.3. Modules/save.py

```

1  print("Module save imported")
2  import pickle
3  import threading
4  import time #Borrar esto
5  import gi
6  import gi.repository
7  gi.require_version('Gtk', '3.0')
8  from gi.repository import Gtk, GObject, Gdk, GdkPixbuf
9
10 glade = "Interface2.glade"
11 last = 0
12 asgl = 1
13
14 ### AUN NO FUNCIONA ###
15
16 def save(allobjects, cabls, aslc=0):
17     global asgl
18     global last
19     if aslc | asgl:
20         asgl = 0
21         sw = loadWindow(mode=1)
22         fil = sw.run()
23         sw.destroy()
24     else:
25         fil = last

```



```

26     if fil != 0:
27         print(fil.split(".")[-1])
28         if fil.split(".")[-1] != "inv":
29             print("Nombre de archivo {} no tiene extensión .inv".format(fil))
30             fil += ".inv"
31         last = fil
32         try:
33             os.remove(fil)
34         except:
35             pass
36         print(allobjects)
37         with open(fil, "wb") as output:
38             pickle.dump((allobjects,cabls), output)
39
40 def load(allobjects, cabls):
41     lw = loadWindow()
42     fil = lw.run()
43     lw.destroy()
44     print(fil)
45     if fil != 0:
46         global last
47         global asgl
48         asgl = 0
49         last = fil
50
51     while len(allobjects) > 0:
52         allobjects[0].delete(pr=0)
53     while len(cabls) > 0:
54         cabls[0].delete()
55     with open(fil, "rb") as inpt:
56         allobj, cables = pickle.load(inpt)
57         dialog = None
58         i = 0
59         st = 1 / ( len(allobj)+len(cables) )
60         fr = 0
61         dialog = loadingdialog(0)
62         def thr():
63             nonlocal dialog
64             dialog = loadingdialog(0)
65
66         print("Thread initiated")
67         dialog.st = 1 / ( len(allobj)+len(cables) )
68         t = threading.Thread(target=thr)
69         t.start()
70         print(allobj)
71         print(cables)
72         lst = []
73         lst.extend(allobj)
74         lst.extend(cables)
75         time.sleep(0.5) #Evitar segfault
76         for obj in lst:
77             dialog.step()
78             obj.load()
79
80         #dialog.destroy()
81
82 class loadWindow(Gtk.Window):
83     def __init__(self, mode=0):
84         self.builder = Gtk.Builder()
85         self.builder.add_from_file(gladefile)
86         self.window = self.builder.get_object("window-filechooser_load")
87         filt = Gtk.FileFilter.new()
88         filt.add_pattern("*.inv")
89         filt.set_name("Archivos .inv")
90         self.window.add_filter(filt)
91         todos = Gtk.FileFilter.new()
92         todos.add_pattern("*")
93         todos.set_name("Todos los tipos de archivo")
94         self.window.add_filter(todos)
95         if mode == 1:

```

```
96         print("Saving")
97         self.window.set_action(Gtk.FileChooserAction.SAVE)
98         self.builder.get_object("window-filechooser-load-this").set_label("Guardar")
99
100     def run(self):
101         rs = self.window.run()
102         self.window.hide()
103         if rs == 1:
104             rs = self.window.get_filename()
105             self.window.destroy()
106             return rs
107     def destroy(self):
108         del self
109
110 class loadingdialog(Gtk.ApplicationWindow):
111     def __init__(self, st):
112         self.builder = Gtk.Builder()
113         self.builder.add_from_file(gladefile)
114         self.window = self.builder.get_object("loading")
115         self.bar = self.builder.get_object("loading-progressbar")
116         self.bar.set_pulse_step(st)
117         GObject.idle_add(self.window.show_all)
118
119         self.st, self.fr = st, 0
120         self.tid = GObject.timeout_add(50, self.tout)
121
122     def step(self):
123         self.fr = self.fr + self.st
124         GObject.idle_add(self.bar.set_fraction, self.fr)
125         print(self.fr)
126
127     def tout(self):
128         self.window.show_all()
129         self.bar.set_fraction(self.fr)
130         return True
131
132     def destroy(self):
133         time.sleep(1)
134         self.window.hide()
135         self.window.destroy()
136         del self
```

This work is licensed under a Creative Commons «Attribution-ShareAlike 4.0 International» license.

