

IES Palas Atenea

Proyecto de Investigación Bachillerato de excelencia

# **Programación, Redes y Código Libre**

---

*David Davó*

Tutor  
Julio Sánchez

1 de septiembre de 2016

# Índice general

<b>1. Programación y código libre</b>	<b>1</b>
1.1. Herramientas . . . . .	1
1.1.1. GNU/Linux . . . . .	1
1.1.2. Git y Github . . . . .	1
1.1.3. LaTeX . . . . .	2
1.1.4. Python . . . . .	2
1.1.5. Gtk+ . . . . .	3
1.1.6. Atom . . . . .	3
1.1.7. Wireshark . . . . .	3
<b>2. Redes Informáticas</b>	<b>4</b>
2.1. Capas de Red/Modelo OSI . . . . .	4
2.2. Elementos físicos de una red . . . . .	5
2.3. Topologías de red . . . . .	5
2.3.1. Clasificación de las topologías de red . . . . .	5
2.3.2. Nodos de una red . . . . .	6
2.3.3. Enlaces de red . . . . .	7
2.4. Paquetes de red . . . . .	7
2.4.1. Ejemplo: Paquete de red . . . . .	8
2.5. Protocolos . . . . .	8
2.5.1. Familia de protocolos de internet . . . . .	8
2.6. Seguridad de redes . . . . .	10
2.6.1. Tipos de ataques . . . . .	10
2.6.2. Contramedidas . . . . .	11
<b>3. El simulador de redes</b>	<b>13</b>
3.1. Instalación . . . . .	13
3.1.1. Ubuntu / Debian . . . . .	13
3.1.2. Arch Linux . . . . .	13
3.1.3. Ejecución manual / instalación portable . . . . .	13
3.2. Uso del programa . . . . .	14
<b>A. Unidades de transferencia de datos</b>	<b>19</b>
<b>B. Código del programa</b>	<b>20</b>
B.1. Main.py . . . . .	20
B.2. Modules/logmod.py . . . . .	51
B.3. Modules/save.py . . . . .	52

# Capítulo 1

## Programación y código libre

Inconsolata(0), bx, n, 10.95  $\square \hookrightarrow \hookrightarrow$   
 $\hookrightarrow$

### 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.

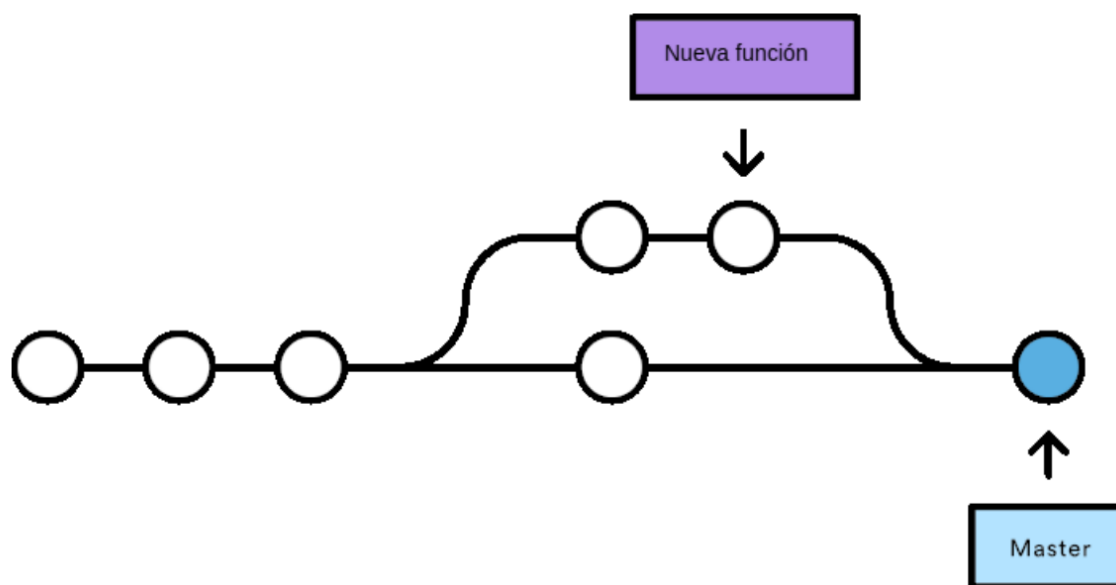
#### 1.1.1. GNU/Linux

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 software son libres y de código abierto. Normalmente Linux se distribuye en distribuciones o 'distros', las cuales contienen paquetes de software preinstalados, dependiendo del grupo de usuarios al que este dirigida.

### Distros

#### 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

Figura 1.1: *Branching* con Git

rama para cada nueva funcionalidad del programa. La implementación del nuevo código a otra rama se denomina *merge*.

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. Tanto el programa como este documento están disponibles en GitHub en el siguiente enlace. <https://github.com/daviddavo/InvProy>

### 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 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*, pero 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 me 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.

### 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.

## Capítulo 2

# Redes Informáticas

### Historia

Internet, tal y como lo conocemos ahora, haciendo uso de IPv6, HTML5, CSS3 no existe 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 multi-nodo. 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. Elementos físicos de una red

Servidor, cliente, switch, hub, router, etc...

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

### 2.3.1. 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.

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

**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.

### 2.3.2. 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. 5), 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...).



### 2.3.3. 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.

## 2.4. Paquetes de red

Un paquete 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 término corrector para el PDU de la capa 2, es un *frame* o marco, y en la capa 4 se denomina segmento o datagrama. 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.

### 2.4.1. Ejemplo: Paquete de red

Para ahorrar espacio, he el paquete en octetos, y estos octetos los he transformado a hexadecimal. Vamos a analizar el paquete de red por el que se ha recibido una página web en HTML. El paquete completo (sin el texto HTML) sería 50465d57cff2f88e855b1ccb0800450004ad848f40003906793f25985812c0a8012a. La primera parte, 50465d57cff2f88e855b1ccb0800, sería la capa de Ethernet. Indicando 48 primeros bits la dirección MAC de destino, y los 48 siguientes, la de origen. Además, los últimos dos octetos (0800) nos indican que se trata de un paquete IPv4.

Ahora, procedemos a analizar la cabecera IP del paquete, 450004ad848f40003906793f25985812c0a8012a. El primer octeto (45) nos indica que se trata de la versión 4 del protocolo IP, y es de 20 bytes de largo. Los dos siguientes octetos indican la longitud total del paquete: 1197 bits.

## 2.5. 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 2.1: Captura de pantalla de Wireshark (Véase 1.1.7, pg. 3) en la que se muestran los protocolos que forman un paquete de red HTTP.

### 2.5.1. 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 millones ( $2^{32}$ ) 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

## 2.6. 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.

### 2.6.1. 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*.

## 2.6.2. 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 descifrar 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 2.2: Captura de pantalla de Wireshark (Véase 1.1.7, pg. 3) 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; phpb3_21j1k_u=1; phpb3_21j1k_k=; phpb3_21j1k_sid=aefe98686186ac00798319aae1ab9be2; style_cookie=null\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: "login" = "index.php"
```

Figura 2.3: 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 debes 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 necesitarás es descargar las dependencias. Esto depende de el 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

**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

**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.

**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 2.3 pág.5)

**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] PSF. *What is Python? Executive Summary*. 2016. url: <https://www.python.org/doc/essays/blurb/>.

# Índice de figuras

1.1. <i>Branching</i> con Git . . . . .	2
2.1. Captura de pantalla de Wireshark . . . . .	8
2.2. Wireshark: SMTP sin encriptación . . . . .	12
2.3. Wireshark: HTTP Form sin encriptación . . . . .	12

## 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("InvProxy 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'")
89     sys.exit()
90
91 try:
92     import cairo
93 except:
94     print("Necesitas tener instalado cairo")
95     print("Como es lógico, pon 'pacman -S python-cairo' en Archlinux")
96     sys.exit()
97
98 #Definiendo un par de cosillas necesarias
99
100 gtk = Gtk
101 config = configparser.RawConfigParser()
102 configdir = "Config.ini"
103 config.read(configdir)
104 allobjects = []
105
106 #Funcion que convierte un numero a una str con [digits] cifras
107 def digitsnumber(number, digits):
108     if len(str(number)) == digits:
109         return str(number)
110     elif len(str(number)) < digits:
111         return "0" * ( digits - len(str(number)) ) + str(number)
112     else:
113         return "-1"
114
115 #Convierte hexadecimal a RGBA tal y como Gdk lo requiere
116 def hex_to_rgba(value):
117     value = value.lstrip('#')
118     if len(value) == 3:

```

```

119         value = ".join([v*2 for v in list(value)])
120         (r1,g1,b1,a1)=tuple(int(value[i:i+2], 16) for i in range(0, 6, 2))+1,)
121         (r1,g1,b1,a1)=(r1/255.00000,g1/255.00000,b1/255.00000,a1)
122
123         return (r1,g1,b1,a1)
124
125     print("#42FF37", hex_to_rgba("#42FF37"))
126
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!!!!11!!")
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 "ERR0R"
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                 ↪ Gtk+" + 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

```



```

187 createlogfile()
188
189 #CONFIGS
190
191 WRES, HRES = int(config.get("GRAPHICS", "WRES")), int(config.get("GRAPHICS", "HRES"))
192 resdir = config.get("DIRS", "respack")
193
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     "on_window1_key_press_event": nothing,
255     "onRestartPress":             restart,
256
257 }
258 builder.connect_signals(handlers)
259
260 builder.get_object("toolbutton1").connect("clicked", objlst.show)
261
262 self.ventana.show_all()
263
264 class Objlst():
265     def __init__(self):
266         self.view = builder.get_object("objetos_treeview")
267         self.tree = Gtk.TreeStore(str, str)
268         renderer = Gtk.CellRendererText()
269         column = Gtk.TreeViewColumn("Objetos", renderer, text=0)
270         self.view.append_column(column)
271         column.set_sort_column_id(0)
272
273         renderer = Gtk.CellRendererText()
274         column = Gtk.TreeViewColumn("Valor", renderer, text=1)
275         column.set_sort_column_id(1)
276         self.view.append_column(column)
277         self.view.set_model(self.tree)
278         self.view.show_all()
279
280         self.revealer = builder.get_object("Revealer1")
281         print("Revealer:", self.revealer.get_reveal_child())
282         self.panpos = 100
283
284     def append(self, obj, otherdata=[]):
285         #SI OBJ YA ESTÁ, QUE AÑADA ATRIBUTOS A LA LISTA.
286         it1 = self.tree.append(None, row=[obj.name, obj.objecttype])
287         it2 = self.tree.append(it1, row=["MAC", str(obj.madir)])
288         itc = self.tree.append(it1, row=["Conexiones", "{}/{ {}".format(len(obj.connections),
289                                     ↳ 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)

```

```

322         self.revealer.set_reveal_child(not self.revealer.get_reveal_child())
323
324         if not rev:
325             pass
326
327     def set_value(self, *args):
328         self.tree.set_value(*args)
329
330     def delete(self, obj):
331         self.tree.remove(obj.trlst)
332
333     def showcfgwindow(self, *args):
334         global configWindow
335         try:
336             configWindow.show()
337         except:
338             configWindow = cfgWindow()
339             configWindow.show()
340
341     #24/06 Eliminada startCable(), incluida en toolbutton_clicked
342
343     def togglegrid(self, *widget):
344         widget = widget[0]
345         global TheGrid
346         obj = TheGrid.backgr_lay
347         if widget.get_active() != True and obj.is_visible():
348             obj.hide()
349         else:
350             obj.show()
351
352     #Una función para gobernarlos a todos.
353     def toolbutton_clicked(self, objeto):
354         global clicked
355         global bttnclicked
356         global areweputtingcable
357         if areweputtingcable != 0:
358             areweputtingcable = 0
359             push_elemento("Cancelada acción de poner un cable")
360
361         if objeto.props.label == "toolbutton5":
362             lprint("Y ahora deberíamos poner un cable")
363             push_elemento("Ahora pulsa en dos objetos")
364             areweputtingcable = "True"
365
366         object_name = objeto.props.label
367         clicked = True
368         bttnclicked = object_name
369
370     #Al pulsar una tecla registrada por la ventana, hace todo esto.
371     def on_key_press_event(self, widget, event):
372         keyname = Gdk.keyval_name(event.keyval).upper() #El upper es por si está BLOQ MAYUS activado.
373         global allkeys #Esta es una lista que almacena todas las teclas que están siendo pulsadas
374         if config.getboolean("BOOLEANS", "print-key-pressed") == True:
375             lprint("Key %s (%d) pulsada" % (keyname, event.keyval))
376             lprint("Todas las teclas: ", allkeys)
377         if not keyname in allkeys:
378             allkeys.add(keyname)
379         if ("CONTROL_L" in allkeys) and ("Q" in allkeys):
380             exiting(1)
381         if ("CONTROL_L" in allkeys) and ("R" in allkeys):
382             restart()
383         if ("CONTROL_L" in allkeys) and ("U" in allkeys):
384             global allobjects
385             print("HARD UPDATE")
386             print(allobjects)
387             for obj in allobjects:
388                 obj.update()
389
390         if ("CONTROL_L" in allkeys) and ("S" in allkeys):
391             global allobjects

```

```

392         MainClase.save()
393     if ("CONTROL_L" in allkeys) and ("L" in allkeys):
394         MainClase.load()
395         allkeys.discard("CONTROL_L")
396         allkeys.discard("L")
397     if ("CONTROL_L" in allkeys) and ("D" in allkeys):
398         theend()
399
400     #Para no tener que hacer click continuamente
401     if ("Q" in allkeys):
402         self.toolbutton_clicked(builder.get_object("toolbutton3"))
403     if "W" in allkeys:
404         self.toolbutton_clicked(builder.get_object("toolbutton4"))
405     if "E" in allkeys:
406         self.toolbutton_clicked(builder.get_object("toolbutton5"))
407     if "R" in allkeys:
408         self.toolbutton_clicked(builder.get_object("toolbutton6"))
409     if "T" in allkeys:
410         self.toolbutton_clicked(builder.get_object("toolbutton7"))
411     return keyname
412
413     #Al dejar de pulsar la tecla deshace lo anterior.
414     def on_key_release_event(self, widget, event):
415         keynameb = Gdk.keyval_name(event.keyval).upper()
416         if config.getboolean("BOOLEANS", "print-key-pressed") == True:
417             lprint("Key %s (%d) released" % (keynameb, event.keyval))
418         global allkeys
419         allkeys.discard(keynameb)
420
421     def drag_drop(widget, context, x, y, time):
422         push_elemento("Drag drop at " + str(x) + "," + str(y) )
423
424     #Comprueba si el objeto tiene una ip asignada
425     def has_ip(self):
426         try:
427             if self.IP != None:
428                 return True
429             else:
430                 return False
431         except:
432             return False
433
434     def save(*args):
435         global cables
436         global allobjects
437         lscl = 0
438         try:
439             if args[1].get_label() == "gtk-save-as":
440                 print("Guardando como")
441                 lscl = 1
442         except:
443             pass
444         save.save(allobjects,cables, aslc=lscl)
445         push_elemento("Guardando...")
446     def load(*args):
447         global cables
448         global allobjects
449         save.load(allobjects,cables)
450         push_elemento("Cargando...")
451     def new(*args):
452         global allobjects
453         global cables
454         save.last = 0
455         while len(allobjects) > 0:
456             allobjects[0].delete(pr=0)
457         while len(cables) > 0:
458             cables[0].delete()
459
460     def new(*args):
461         global cables

```

```

462     global allobjects
463     while len(allobjects) > 0:
464         allobjects[0].delete(pr=0)
465
466     #Esta clase no es mas que un prompt que pide 'Si' o 'No'.
467     #La función run() retorna 1 cuando se clicka si y 0 cuando se clicka no, así sirven como enteros y booleans.
468     class YesOrNoWindow(Gtk.Dialog):
469         def __init__(self, text, *args, Yest="Sí", Not="No"):
470
471             self.builder = Gtk.Builder()
472             self.builder.add_from_file(gladefile)
473
474             self.yesornowindow = self.builder.get_object("YesOrNoWindow")
475             self.labeldialog = self.builder.get_object("YoN_label")
476             self.nobutton = self.builder.get_object("YoN_No")
477             self.yesbutton = self.builder.get_object("YoN_Yes")
478
479             self.nobutton.connect("clicked", self.on_button_clicked)
480             self.yesbutton.connect("clicked", self.on_button_clicked)
481
482             self.labeldialog.set_text(text)
483             self.yesbutton.set_label(Yest)
484             self.nobutton.set_label(Not)
485
486             self = self.yesornowindow
487
488         def on_button_clicked(self, widget):
489             dialog = self
490
491         def run(self):
492             return self.yesornowindow.run()
493             self.yesornowindow.hide()
494
495         def destroy(self):
496             self.yesornowindow.destroy()
497
498     objetocable1 = None
499
500     #Esto es el Grid donde van las cosicas. A partir de aqui es donde esta lo divertido.
501     class Grid():
502         def __init__(self):
503             #16/06/16 MAINPORT PASA A SER VARIAS LAYERS
504             self.overlay = builder.get_object("overlay1")
505             self.mainport = Gtk.Layout.new()
506             self.cables_layer = Gtk.Layout.new()
507             self.backgr_layer = Gtk.Layout.new()
508             self.select_layer = Gtk.Layout.new() #Aparecer un fondo naranja en la cuadrícula cuando se selecciona un objeto
509             self.animat_layer = Gtk.Layout.new() #La capa de las animaciones de los cables
510             self.overlay.add_overlay(self.backgr_layer)
511             self.overlay.add_overlay(self.select_layer)
512             self.overlay.add_overlay(self.cables_layer)
513             self.overlay.add_overlay(self.animat_layer)
514             self.overlay.add_overlay(self.mainport)
515
516             self.viewport = builder.get_object("viewport1")
517             self.eventbox = builder.get_object("eventbox1")
518             self.eventbox.connect("button-press-event", self.clicked_on_grid)
519             #self.viewport.get_hadjustment().set_value(800)
520
521             self.wres = config.getint("GRAPHICS", "viewport-wres")
522             self.hres = config.getint("GRAPHICS", "viewport-hres")
523             self.sqres = config.getint("GRAPHICS", "viewport-sqres")
524             self.overlay.set_size_request(self.wres*self.sqres, self.hres*self.sqres)
525
526             #Modifica el color de fondo del viewport
527             clr = hex_to_rgba(config.get("GRAPHICS", "viewport-background-color"))
528             print("CLR:", clr)
529             self.viewport.override_background_color(Gtk.StateFlags.NORMAL, Gdk.RGBA(*clr))
530
531             #13/07/16 Ahora esto va por cairo, mejooooor.

```

```

532     ### INICIO CAIRO
533
534     width, height, sq = self.wres*self.sqres, self.hres*self.sqres, self.sqres
535     surface = cairo.ImageSurface(cairo.FORMAT_ARGB32, width, height)
536     ctx = cairo.Context(surface)
537     ctx.close_path ()
538     ctx.set_source_rgba(0,0,0,1)
539     ctx.set_line_width(1)
540
541     for i in range(self.wres):
542         ctx.move_to(i*sq, 0)
543         ctx.line_to(i*sq, height)
544     for i in range(self.hres):
545         ctx.move_to(0, i*sq)
546         ctx.line_to(width, i*sq)
547
548
549     ctx.stroke()
550     self.image = Gtk.Image.new_from_surface(surface)
551     ### FINAL DE LO DE CAIRO
552
553     self.backgr_lay.put(self.image, 0, 0)
554
555     def subshow(widget):
556         #Para que no aparezca arriba a la izquierda:
557         scrolled = builder.get_object("scrolledwindow1")
558         scrolled.get_vadjustment().set_value(height/3)
559         scrolled.get_hadjustment().set_value(width/3)
560
561     if config.getboolean("GRAPHICS", "start-centered"):
562         builder.get_object("window1").connect("show", subshow)
563     self.overlay.show_all()
564     self.contadorback = 0
565
566     def moveto(self, image, x, y, *args, layout=None):
567         if x < self.wres and y < self.hres:
568             if layout == None:
569                 layout = self.mainport
570             elif str(layout.__class__.__name__) == "Layout":
571                 layout = layout
572             else:
573                 print("layout.__class__.__name__", layout.__class__.__name__)
574             if image in layout.get_children():
575                 layout.move(image, x*self.sqres, y*self.sqres)
576             else:
577                 layout.put(image, x*self.sqres, y*self.sqres)
578         else:
579             print("\033[31mError: Las coordenadas se salen del grid\033[00m")
580
581     def clicked_on_grid(self, widget, event, *args):
582         global clicked
583         global bttnclicked
584         global allobjects
585         global areweputtingcable
586         self.contadorback += 1
587
588         push_elemento("Clicked on grid @" + str(self.gridparser(event.x, self.wres)) + "," +
589             ↵ str(self.gridparser(event.y, self.hres)))
590
591         if self.searchforobject(self.gridparser(event.x, self.wres), self.gridparser(event.y, self.hres)) == False:
592             if clicked == 1:
593                 push_elemento("Clicked: " + str(clicked) + " bttnclicked: " + str(bttnclicked))
594                 if bttnclicked == "Router":
595                     Router(self.gridparser(event.x, self.wres), self.gridparser(event.y, self.hres))
596                     push_elemento("Creado objeto router")
597                 elif bttnclicked == "toolbutton4":
598                     Switch(self.gridparser(event.x, self.wres), self.gridparser(event.y, self.hres))
599                     push_elemento("Creado objeto switch")
600                 elif bttnclicked == "toolbutton6":
601                     Computador(self.gridparser(event.x, self.wres), self.gridparser(event.y, self.hres))

```

```

601         push_elemento("Creado objeto Computador")
602     elif btnnclicked == "toolbutton7":
603         Hub(self.gridparser(event.x, self.wres), self.gridparser(event.y, self.hres))
604         push_elemento("Creado objeto Hub")
605
606 elif self.searchforobject(self.gridparser(event.x, self.wres), self.gridparser(event.y, self.hres)) != False:
607     push_elemento("Ahí ya hay un objeto, por favor selecciona otro sitio")
608 else:
609     lprint("pls rebisa l codigo")
610     clicked = 0
611     btnnclicked = 0
612
613 #Button: 1== Lclick, 2== Mclick
614 #Para comprobar si es doble o triple click: if event.type == gtk.gdk.BUTTON_PRESS, o gtk.gdk_2_BUTTON_PRESS
615 if event.button == 3:
616     rclick_Object = self.searchforobject(self.gridparser(event.x, self.wres), self.gridparser(event.y,
617         ↪ self.hres))
618     if rclick_Object != False:
619         rclick_Object.rclick(event)
620     else:
621         print("Agua")
622
623 if areweputtingcable != 0:
624     objeto = self.searchforobject(self.gridparser(event.x, self.wres), self.gridparser(event.y, self.hres))
625     if objeto == False:
626         push_elemento("Selecciona un objeto por favor")
627     elif objeto != False:
628         if len(objeto.connections) < objeto.max_connections:
629             if areweputtingcable == "True":
630                 push_elemento("Ahora selecciona otro más")
631                 areweputtingcable = "Secondstep"
632                 global objetocable1
633                 objetocable1 = objeto
634             elif areweputtingcable == "Secondstep":
635                 push_elemento("Poniendo cable")
636                 areweputtingcable = 0
637                 global objetocable1
638                 cable = Cable(objetocable1, objeto)
639                 objeto.connect(objetocable1, cable)
640                 objetocable1 = 0
641         else:
642             push_elemento("Número máximo de conexiones alcanzado")
643
644 #Te pasa las cordenadas int que retorna Gtk a coordenadas del Grid, bastante sencillito. Tienes que llamarlo 2
645 ↪ veces, una por coordenada
646 def gridparser(self, coord, cuadrados, mode=0):
647     if mode == 0:
648         partcoord = coord / self.sqres
649         for i in range(cuadrados + 1):
650             if partcoord < i:
651                 return i
652             else:
653                 pass
654     if mode == 1:
655         return coord * self.sqres
656
657 def resizetogrid(self, image):
658     #Image debe ser una imagen gtk del tipo gtk.Image
659     pixbuf = image.get_pixbuf()
660     pixbuf = pixbuf.scale_simple(self.sqres, self.sqres, GdkPixbuf.InterpType.BILINEAR)
661     image.set_from_pixbuf(pixbuf)
662
663 #Una función para encontrarlos,
664 def searchforobject(self, x, y):
665     global allobjects
666     localvar = False
667     for i in range(len(allobjects)):
668         if allobjects[i].x == x:
669             if allobjects[i].y == y:

```

```

669         localvar = True
670         objeto = allobjects[i]
671         break
672     if localvar == True:
673         return objeto
674     else:
675         return False
676
677     def __str__(self):
678         lprint("No se que es esto")
679
680 TheGrid = Grid()
681
682 #Clases de los distintos objetos. Para no escribir demasiado tenemos la clase ObjetoBase
683 #De la que heredaran las demas funciones
684 cnt_objects = 1
685 cnt_rows = 2
686 objlst = MainClase.ObjLst()
687
688 import uuid
689
690 class ObjetoBase():
691     allobjects = []
692     cnt = 0
693     #Una función para atraerlos a todos y atarlos en las tinieblas
694     def __init__(self, x, y, objtype, *args, name="Default", maxconnections=4, ip=None):
695         global cnt_objects
696         global cnt_rows
697         global allobjects
698         global gladefile
699
700         #IMPORTANTE: GENERAR UUID PARA CADA OBJETO
701         #La v4 crea un UUID de forma aleatoria
702         self.uuid = uuid.uuid4()
703         print("\033[96mUUID:\033[00m", self.uuid)
704
705         self.builder = Gtk.Builder()
706         self.builder.add_from_file(gladefile)
707         self.menuemergente = self.builder.get_object("grid_rclick")
708         self.builder.get_object("grid_rclick-disconnect_all").connect("activate", self.disconnect)
709         self.builder.get_object("grid_rclick-delete").connect("activate", self.delete)
710         self.builder.get_object("grid_rclick-debug").connect("activate", self.debug)
711
712         allobjects.append(self)
713
714         self.realx = x * TheGrid.sqres
715         self.realy = y * TheGrid.sqres
716         self.x = x -1
717         self.y = y -1
718         self.connections = []
719         self.cables = []
720         self.max_connections = maxconnections
721
722         #Algún día pasaré todos los algoritmos a algoritmos de búsqueda binaria
723         for f in os.listdir(resdir):
724             lprint(f, f.startswith(objtype))
725             if f.startswith(objtype) and ( f.endswith(".jpg") or f.endswith(".png") ):
726                 self.imgdir = resdir + f
727                 break
728
729         self.image = gtk.Image.new_from_file(self.imgdir)
730         self.resizetogrid(self.image)
731         if name == "Default" or name == None:
732             self.name = self.objtype + " " + str(self.__class__.cnt)
733         else:
734             self.name = name
735         cnt_objects += 1
736         self.__class__.cnt += 1
737
738         TheGrid.moveto(self.image, self.x, self.y)

```



```

739     self.image.show()
740
741     self.maddir = self.mac()
742     print("MAC:", self.maddir, int(self.maddir), bin(self.maddir))
743     if ip == None:
744         print("No ip definida")
745         self.ipstr = "None"
746
747     #Ahora vamos con lo de aparecer en la lista de la izquierda,
748     #aunque en realidad es un grid
749     lista = objlst
750     self.trlst = lista.append(self)
751     self.image.set_tooltip_text(self.name + " (" + str(len(self.connections)) + "/" + str(self.max_connections)
752     ↪ + ")\" + self.ipstr)
753
754     self.window_changethings = w_changethings(self)
755     self.builder.get_object("grid_rclick-name").connect("activate", self.window_changethings.show)
756
757     self.cnt = 0 #Se me olvido que hace esta cosa
758
759 def load(self):
760     global cnt_objects
761     global cnt_rows
762     global allobjects
763     self.builder = Gtk.Builder()
764     self.builder.add_from_file(gladefile)
765     self.builder.get_object("grid_rclick").connect("activate", self.disconnect)
766     self.builder.get_object("grid_rclick-delete").connect("activate", self.delete)
767     self.builder.get_object("grid_rclick-debug").connect("activate", self.debug)
768     self.connections = []
769     self.cables = []
770     cnt_objects += 1
771     self.__class__.cnt += 1
772     allobjects.append(self)
773     self.image = gtk.Image.new_from_file(self.imgdir)
774     self.resizetogrid(self.image)
775     TheGrid.moveto(self.image, self.x-1, self.y-1)
776     self.image.show()
777     lista = builder.get_object("grid2")
778     lista.insert_row(cnt_rows)
779     self.label = Gtk.Label.new(self.name)
780     lista.attach(self.label, 0, cnt_rows, 1, 1)
781     cnt_rows += 1
782     self.label.show()
783     self.image.set_tooltip_text(self.name + " (" + str(len(self.connections)) + "/" + str(self.max_connections)
784     ↪ + ")\" + self.ipstr)
785     self.window_changethings = w_changethings(self)
786     self.builder.get_object("grid_rclick-name").connect("activate", self.window_changethings.show)
787
788     print("CABLES",self.cables)
789
790 #Esta funcion retorna una str cuando se usa el objeto. En lugar de <0XXXXXXXX object>
791 def __str__(self):
792     return "<Tipo: " + self.objecttype + " | Name: " + self.name + " | Pos: " + str(self.x) + ", " + str(self.y) +
793     ↪ ">"
794
795 def debug(self, *args):
796     print("DEBUG")
797     print("MAC:", self.maddir, int(self.maddir))
798
799 def rclick(self, event):
800     global rclick_Object
801     rclick_Object = self
802
803     print(self)
804     lprint("rclick en", self.x, self.y, self.objecttype, "\nConnections: ", end="")
805     lprint(self.connections)
806     self.rmenu = self.menuemergente
807     if self.objecttype == "Computer" and len(self.compcon()) > 0:

```

```

806         self.builder.get_object("grid_rclick-sendpkg").show()
807     else:
808         self.builder.get_object("grid_rclick-sendpkg").hide()
809     if len(self.connections) > 0:
810         self.builder.get_object("grid_rclick-disconnect").show_all()
811     else:
812         self.builder.get_object("grid_rclick-disconnect").hide()
813     self.rmenu.popup(None, None, None, None, event.button, event.time)
814
815 def resizetogrid(self, image, *args):
816     #Ver resizetogrid en Grid (clase)
817     lprint(*args)
818     TheGrid.resizetogrid(image)
819
820 def clickado(self, widget, event):
821     lprint("Clickado en objeto " + str(self) + " @ " + str(self.x) + ", " + str(self.y))
822
823 class mac():
824     def __init__(self, *macaddr, bits=48):
825         print("macaddr:", *macaddr)
826         if macaddr == None or True:
827             tmp = self.genmac(self, bits=bits)
828
829             self.int = tmp[0]
830             self.str = tmp[1]
831             self.bin = ("{:0:0}"+str(bits)+"b").format(self.int)
832
833     def genmac(*self, bits=48, mode=None):
834         #Por defecto se usa mac 48, o lo que es lo mismo, la de toa la vida
835         #Nota, falta un comprobador de que la mac no se repita
836         reallmac = int("11" + str("{0:0}" + str(bits-2) + "b").format(random.getrandbits(bits-2)),2)
837         readmac = str(hex(reallmac)).upper().replace("0X", "")
838         readmac = ":".join([readmac[i * 2:i * 2 + 2] for i,bl in enumerate(readmac[:2])])
839         if mode == 0:
840             return reallmac
841         if mode == 1:
842             return readmac
843         else:
844             return [reallmac, readmac]
845
846     def __str__(self):
847         readmac = str(hex(self.int)).upper().replace("0X", "")
848         return ":".join([readmac[i * 2:i * 2 + 2] for i,bl in enumerate(readmac[:2])])
849
850     def __bytes__(self):
851         return Object.__bytes__(self)
852
853     def __int__(self):
854         return self.int
855     def __index__(self):
856         return self.int
857     def list(self):
858         return self.str.split(":")
859
860
861 #Esta función se encarga de comprobar a que ordenador(es) está conectado
862 #en total, pasando por routers, hubs y switches.
863
864 #Nota, hacer que compruebe que ordenadores tienen IP, y cuales no.
865 def compcon(self, *args):
866     passedyet = []
867     comps = []
868     reself = self
869
870     def subcompcon(notself, *args):
871         nonlocal passedyet
872         nonlocal reself
873         subcomps = []
874
875         interc = notself.connections

```

```

876         #print(notself, "connections:", iterc)
877         #next(iterc)
878
879         for con in iterc:
880             if con.uuid != reself.uuid and con.uuid not in [obj.uuid for obj in passedyet]:
881                 passedyet.append(con)
882                 #print(con)
883                 if con.objecttype == "Computer":
884                     subcomps.append(con)
885                 elif con.objecttype == "Switch" or con.objecttype == "Hub":
886                     subcomps.extend(subcompcon(con))
887                 else:
888                     print("Saltado", con)
889                     pass
890                 #passedyet.append(con)
891
892         #print("passedyet", passedyet)
893         return subcomps
894
895     comps.extend(subcompcon(self))
896
897     try:
898         #comps.remove(self)
899         pass
900     except:
901         pass
902
903     if args == 1 or "Gtk" in str(args):
904         print("Comps:", comps)
905         print("\nCompsname:", [x.name for x in comps])
906
907     return comps
908
909     #Comprueba si un objeto está conectado a otro.
910     def isconnected(self, objeto):
911         cons = compcon(self)
912         if objeto in cons:
913             return True
914         else:
915             return False
916
917     #TODO: Para no tener que actualizar todo, que compruebe el que cambió
918     #TODO: !! Hacer que modifique el menu_emergente (Hecho a medias xds)
919     #Nota !!: No puedes buscar un objeto en una lista, debes buscar sus atr.
920     def update(self):
921         print("\033[95m>>Updating\033[00m", self)
922         print(self.builder.get_object("grid_rclick-disconnect"))
923         self.image.set_tooltip_text(self.name + " (" + str(len(self.connections)) + "/" + str(self.max_connections)
924                                     + ")")
925
926         objlst.set_value(self.trlst, 0, self.name)
927
928         objlst.update(self, "MAC", str(self.macdir))
929         for child in self.builder.get_object("grid_rclick-disconnect").get_submenu().get_children():
930             if child.props.label.upper() != "TODOs":
931                 if child.link.uuid not in [x.uuid for x in self.connections]:
932                     print("Object", child.link.__repr__(), "in connections", self.connections)
933                     child.hide()
934                     child.destroy()
935                 else:
936                     print("Object", child.link.__repr__(), "in self.connections", self.connections)
937             pass
938
939         objlst.upcon(self)
940
941         print("\033[95m<<\033[00m")
942
943     def connect(self, objeto, cable):
944         tmp = Gtk.MenuItem.new_with_label(objeto.name)
945         self.builder.get_object("grid_rclick-disconnect").get_submenu().append(tmp)
946         tmp.show()

```

```

945     tmp.connect("activate", self.disconnect)
946     #link es un objeto vinculado al widget, luego es útil.
947     tmp.link = objeto
948     tmp2 = Gtk.MenuItem.new_with_label(objeto.name)
949     self.builder.get_object("grid_rclick-sendpkg").get_submenu().append(tmp2)
950     if self.__class__.__name__ != "Switch" and self.__class__.__name__ != "Hub":
951         tmp2.connect("activate", self.send_pck)
952         tmp2.show()
953     tmp2.link = objeto
954
955     tmp = Gtk.MenuItem.new_with_label(self.name)
956     objeto.builder.get_object("grid_rclick-disconnect").get_submenu().append(tmp)
957     tmp.show()
958     tmp.connect("activate", objeto.disconnect)
959     tmp.link = self
960     tmp2 = Gtk.MenuItem.new_with_label(self.name)
961     objeto.builder.get_object("grid_rclick-sendpkg").get_submenu().append(tmp2)
962     if objeto.__class__.__name__ != "Switch" and objeto.__class__.__name__ != "Hub":
963         tmp2.show()
964         tmp2.connect("activate", objeto.send_pck)
965     tmp2.link = self
966
967     self.connections.append(objeto)
968     self.cables.append(cable)
969     #objlst.tree.append(self.trdic["Connections"], row=[objeto.name, objeto.objectype])
970
971     objeto.connections.append(self)
972     objeto.cables.append(cable)
973     #objlst.tree.append(objeto.trdic["Connections"], row=[self.name, self.objectype])
974
975     self.update()
976     objeto.update()
977
978     if objeto.__class__.__name__ == "Switch":
979         print("Connecting {} to {}".format(objeto, self))
980         objeto.connectport(self)
981     if self.__class__.__name__ == "Switch":
982         print("Connecting {} to {}".format(objeto, self))
983         self.connectport(objeto)
984
985     def disconnect(self, widget, *args, de=None):
986         print("Cables:", self.cables)
987         #QUICKFIX
988         try:
989             if widget.props.label.upper() == "TODOS" and de == None:
990                 de = "All"
991             elif de == None:
992                 de = widget.link
993         except:
994             print("NO WIDGET AT DISCONNECT()")
995
996         if de == "All":
997             ###NO FUNCIONA DEL TODO BIEN, NO USAR###
998             #Bug, el ultimo cable no se borra
999             print("Ahora a desconectar de todos")
1000             while len(self.connections) > 0:
1001                 self.disconnect(widget, de=self.connections[0])
1002
1003         else:
1004             objlst.tree.remove(self.trcondic[de])
1005             del self.trcondic[de]
1006             objlst.tree.remove(de.trcondic[self])
1007             del de.trcondic[self]
1008
1009             de.connections.remove(self)
1010             self.connections.remove(de)
1011
1012             iterc = iter(self.builder.get_object("grid_rclick-disconnect").get_submenu().get_children())
1013             next(iterc)
1014             print("\033[91mLinks\033[00m", [x.link for x in iterc])

```

```

1015         if de in [x.link for x in interc]:
1016             print("\033[91mSelf in\033[00m", self)
1017
1018         for cable in self.cables:
1019             if cable.fromobj == self or cable.toobj == self:
1020                 cable.delete()
1021                 break
1022
1023         de.update()
1024
1025         if self.__class__.__name__ == "Switch":
1026             self.disconnectport(de)
1027         elif de.__class__.__name__ == "Switch":
1028             de.disconnectport(self)
1029
1030     self.update()
1031
1032 def delete(self, *widget, conf=1, pr=1):
1033     if pr == 1:
1034         yonW = YesOrNoWindow("¿Estás seguro de que quieres eliminar " + self.name + " definitivamente? El objeto
1035             ↪ será imposible de recuperar y te hechará de menos.")
1036         yonR = yonW.run()
1037         yonW.destroy()
1038     else:
1039         yonR = 1
1040     if yonR == 1:
1041         self.disconnect(0, de="All")
1042         objlst.delete(self)
1043         self.image.destroy()
1044         global allobjects
1045         allobjects.remove(self)
1046     elif yonR == 0:
1047         print("Piénsatelo dos veces")
1048     else:
1049         raise
1050
1051 def packet_received(self, pck, *args, port=None):
1052     print("Hola, soy {} y he recibido un paquete, pero no sé que hacer con él".format(self.name))
1053     if config.getboolean("DEBUG", "packet-received"):
1054         print(">Pck:", pck)
1055         if pck.frame != None:
1056             print("\033[91m>>Atributos del paquete\033[00m")
1057             totalen = pck.lenght + 14*8
1058             wfr = bformat(pck.frame, (totalen+14)*8)
1059             print(">Wfr:", wfr)
1060             mac1 = "{0:011b}".format(pck.frame)[0:6*8]
1061             print(">Mac:", int(mac1,2))
1062             readmac = str(hex(int(mac1,2))).strip("0x")
1063             print(":".join([readmac[i * 2:i * 2 + 2] for i,bl in enumerate(readmac[:2])]).upper())
1064
1065             print("<<Fin de los atributos")
1066
1067 npack = 0
1068
1069 class Router(ObjetoBase):
1070     cnt = 1
1071     def __init__(self, x, y, *args, name="Default"):
1072         global cnt_objects
1073         self.objecttype = "Router"
1074         push_elemento("Creado Objeto Router")
1075
1076         ObjetoBase.__init__(self, x, y, self.objecttype, name=name)
1077         self.x = x
1078         self.y = y
1079
1080     def __del__(self, *args):
1081         push_elemento("Eliminado objeto")
1082         del self
1083

```

```

1084 class Switch(ObjetoBase):
1085     cnt = 1
1086     #El objeto puerto
1087     class Port():
1088         def __init__(self, switch):
1089             self.id = switch.portid
1090             self.dic = switch.pdic
1091             self.all = switch.pall
1092             switch.portid += 1
1093             self.switch = switch
1094             self.connection = None
1095             self.all[self.id] = self
1096             self.dic[self.id] = self.connection
1097         def connect(self, connection):
1098             self.connection = connection
1099             self.dic[self.id] = self.connection
1100         def disconnect(self):
1101             self.connection = None
1102             self.dic[self.id] = self.connection
1103         def is_available(self):
1104             if self.connection == None:
1105                 return True
1106             return False
1107
1108     class w_switch_table(Gtk.ApplicationWindow):
1109         def __init__(self, switch):
1110             self.link = switch
1111             builder = switch.builder
1112             builder.get_object("window_switch-table-button").connect("clicked", self.hide)
1113             builder.get_object("window_switch-table").connect("delete-event", self.hide)
1114             self.store = Gtk.ListStore(str,int,int,int)
1115
1116             self.view = builder.get_object("window_switch-table-TreeView")
1117             self.view.set_model(self.store)
1118             for i, column_title in enumerate(["MAC", "Puerto", "TTL (s)"]):
1119                 renderer = Gtk.CellRendererText()
1120                 column = Gtk.TreeViewColumn(column_title, renderer, text=i)
1121                 column.set_sort_column_id(i)
1122                 self.view.append_column(column)
1123             self.ticking = False
1124             builder.get_object("window_switch-table").set_keep_above(True)
1125
1126         def show(self, *a):
1127             self.ticking = True
1128             GObject.timeout_add(1001, self.tick)
1129             for row in self.store:
1130                 row[2] = row[3] - time.time()
1131             self.link.builder.get_object("window_switch-table").show_all()
1132
1133         def hide(self, window, *event):
1134             self.link.builder.get_object("window_switch-table").hide()
1135             self.ticking = False
1136             return True
1137         def append(self, lst):
1138             lst.append(lst[2])
1139             for row in self.store:
1140                 row[2] = row[3] - time.time()
1141             print(lst)
1142             row = self.store.append(lst)
1143             print(self.view.get_property("visible"))
1144             if self.view.get_property("visible") == True:
1145                 self.ticking = True
1146                 GObject.timeout_add(1001, self.tick)
1147
1148         def tick(self):
1149             for row in self.store:
1150                 row[2] = row[3] - time.time()
1151                 if row[2] <= 0:
1152                     try:
1153                         self.store.remove(row.iter)

```

```

1154         self.link.table.remove(row)
1155     except:
1156         pass
1157     if len(self.store) == 0:
1158         self.ticking = False
1159     return self.ticking
1160 def remove(self, lst):
1161     for row in self.store:
1162         if row == lst:
1163             self.store.remove(row.iter)
1164             self.link.table
1165             break
1166     pass
1167
1168 def __init__(self, x, y, *args, name="Default", maxconnections=5):
1169     self.objecttype = "Switch"
1170     self.portid = 0
1171     self.pdic = {}
1172     self.pall = {}
1173
1174     push_elemento("Creado objeto Switch")
1175     self.imgdir = resdir + "Switch.*"
1176     ObjetoBase.__init__(self, x, y, self.objecttype, name=name, maxconnections=maxconnections)
1177     self.x = x
1178     self.y = y
1179     self.timeout = 20 #Segundos
1180
1181     for p in range(self.max_connections):
1182         self.Port(self)
1183     print(self.pall)
1184
1185     self.table = [
1186         #[MAC, port, expiration]
1187     ]
1188     self.wtable = self.w_switch_table(self)
1189     child = Gtk.MenuItem.new_with_label("Routing Table")
1190     self.builder.get_object("grid_rclick").append(child)
1191     child.connect("activate", self.wtable.show)
1192     child.show()
1193
1194     self.ch = child
1195
1196 def load(self):
1197     ObjetoBase.load(self)
1198     del self.wtable
1199     self.table = []
1200     self.wtable = self.w_switch_table(self)
1201
1202     del self.ch
1203     child = Gtk.MenuItem.new_with_label("Routing Table")
1204     self.builder.get_object("grid_rclick").append(child)
1205     child.connect("activate", self.wtable.show)
1206     child.show()
1207
1208     self.ch = child
1209
1210
1211 def connectport(self, objeto):
1212     for port in self.pall:
1213         if self.pall[port].is_available():
1214             self.pall[port].connect(objeto)
1215             break
1216     print(self.pdic)
1217
1218 def disconnectport(self, objeto):
1219     for p in self.pdic:
1220         print("i: {}, idx: {}".format(p, self.pdic[p]))
1221         if objeto == self.pdic[p]:
1222             self.pall[p].disconnect()
1223             break

```

```

1224         print(self.pdic)
1225
1226     def packet_received(self, pck, port=None):
1227         macd = "{0:0112b}".format(pck.frame)[0:6*8]
1228         macs = "{0:0112b}".format(pck.frame)[6*8+1:6*16+1]
1229
1230         #LO PRIMERO: AÑADIRLO A LA TABLA
1231         readmac = str(hex(int(macs,2))).upper().replace("0X", "")
1232         readmac = ":".join([readmac[i * 2:i * 2 + 2] for i,b1 in enumerate(readmac[:2])])
1233
1234         for tab in self.table:
1235             if tab[2] <= time.time():
1236                 print("Ha llegado tu hora")
1237                 self.table.remove(tab)
1238                 self.wtable.remove(tab)
1239             if tab[0] == int(macd,2):
1240                 print("TAB[0] == mcd")
1241                 tab[2] = int(time.time()+self.timeout)
1242                 for row in self.wtable.store:
1243                     print(row[0], tab[0])
1244                     if int(row[0].replace(":", ""),16) == tab[0]:
1245                         row[3] = int(time.time()+self.timeout)
1246         if int(macs,2) not in [x[0] for x in self.table]:
1247             tmp = [int(macs,2), port, int(time.time()+self.timeout)]
1248             self.table.append(tmp)
1249             tmp = [readmac, port, int(time.time()+self.timeout)]
1250             self.wtable.append(tmp)
1251
1252         #####
1253
1254         #ObjetoBase.packet_received(self, pck)
1255
1256         ttl = int(pck.str[64:72],2)
1257         ttlnew = "{0:08b}".format(ttl-1)
1258         pck.str = "".join(( pck.str[:64], ttlnew, pck.str[72:] ))
1259
1260         print("self.macdir",int(self.macdir), int("{0:0112b}".format(pck.frame)[6*8+1:6*16+1],2))
1261         print("TTL:", int(pck.str[64:72],2), pck.str[64:72])
1262
1263         print("Soy {} y mi deber es entregar el paquete a {}".format(self.name,int(macd,2)))
1264         print("El paquete llegó por el puerto {}".format(port))
1265         dic = {}
1266         for i in self.connections:
1267             dic[int(i.macdir)] = i
1268         print("Connections MAC's:", dic)
1269
1270         #Cambiamos los bits de macs
1271         #Si macd en conn, enviarle el paquete
1272         #Si existe una tabla de enrutamiento que contiene una ruta para macd, enviar por ahi
1273         #Si no, enviar al siguiente, y así
1274         print(">MAAAC:",int(macd,2), "DIIIC:")
1275         if int(macd,2) in dic and ttl > 0:
1276             pck.animate(self, dic[int(macd,2)])
1277
1278         elif int(macd,2) in [x[0] for x in self.table]:
1279             for x in self.table:
1280                 if x[0] == int(macd,2):
1281                     pck.animate(self, self.pdic[x[1]])
1282
1283         elif "Switch" in [x.objectype for x in self.connections] and ttl >= 0:
1284             print("Ahora lo enviamos al siguiente router")
1285             print(int(macd,2), dic)
1286             tmp1st = self.connections[:] #Crea una nueva copia de la lista
1287             print(tmp1st)
1288             for i in tmp1st:
1289                 if int(macs,2) == int(i.macdir):
1290                     print("REMOVING", i)
1291                     tmp1st.remove(i)
1292             try:
1293                 tmp1st.remove(*[x for x in tmp1st if x.objectype == "Computer"])

```



```

1294         except TypeError:
1295             pass
1296         print("Tmplst:", tmplst)
1297         obj = choice(tmplst)
1298         print("Sending to:", obj)
1299         pck.animate(self, obj)
1300
1301     def debug(self, *args):
1302         print(self.pdic)
1303         print("MyMac:", self.macdir)
1304         row_format = "{:>20}" * 3
1305         print(row_format.format("MAC", "NXT", "EXP s"))
1306         for row in self.table:
1307             if row[1] == None:
1308                 row[1] = "None"
1309             if int(row[2]-time.time()) <= 0:
1310                 self.table.remove(row)
1311             print(row_format.format(row[0], row[1], int(row[2]-int(time.time()))))
1312
1313     #¿Tengo permisos de escritura?, no se si tendré permisos
1314     #Update: Si los tenía
1315     class Hub(ObjetoBase):
1316         cnt = 1
1317         def __init__(self, x, y, *args, name="Default", maxconnections=4, ip=None):
1318             self.objecttype = "Hub"
1319             push_elemento("Creado objeto Hub")
1320             self.imgdir = resdir + "Hub.*"
1321             ObjetoBase.__init__(self, x, y, self.objecttype, name=name)
1322             self.x = x
1323             self.y = y
1324
1325         def packet_received(self, pck, port=None):
1326             ttl = int(pck.str[64:72], 2)
1327             macs = "{0:012b}".format(pck.frame)[6*8+1:6*16+1]
1328             ttlnew = "{0:08b}".format(ttl-1)
1329             pck.str = "".join(( pck.str[:64], ttlnew, pck.str[72:] ))
1330             if ttl >= 0:
1331                 for obj in self.connections:
1332                     pck.animate(self, obj)
1333
1334     class Computador(ObjetoBase):
1335         cnt = 1
1336         def __init__(self, x, y, *args, name="Default", maxconnections=1, ip=None):
1337             self.objecttype = "Computer"
1338
1339             push_elemento("Creado objeto Computador")
1340             self.img = resdir + "Comp.*"
1341             ObjetoBase.__init__(self, x, y, self.objecttype, name=name)
1342             self.x = x
1343             self.y = y
1344             self.max_connections = maxconnections
1345             self.IP = None
1346             self.update()
1347
1348     class ip():
1349         def __init__(self, *args, ipstr="None"):
1350             self.str = ipstr
1351
1352         def __str__(self):
1353             return self.str
1354
1355         def set_str(self, str):
1356             self.str = str
1357             self.parser(str, 0)
1358
1359         def set_bin(self, binar):
1360             t = binar
1361             print(bin(t))
1362             if "\0b" not in str(t) and "." in str(t):
1363                 print("Type is str")

```

```

1364         self.bins = t
1365     elif "0b" in str(bin(t)) and "." not in str(bin(t)):
1366         print("Type is binar")
1367         self.bin = t
1368     else:
1369         print("Error:", t)
1370     self.parser(t, 1)
1371
1372 #ip2p stands 4 'ip to parse'
1373 def parser(self, ip2p, mode):
1374     #mode 0: str2b
1375     if mode == 0:
1376         tmp1st = ip2p.split(".")
1377         toreturn = []
1378         for i in tmp1st:
1379             i = int(i)
1380             toreturn.append("{0:08b}".format(i))
1381         self.bins = ".".join(toreturn)
1382         self.bin = int(self.bins.replace(".", ""), base=2)
1383         return self.bins
1384
1385     #mode 1: b2str
1386     elif mode == 1:
1387         if "0b" not in str(ip2p):
1388             self.bin = bin(int(ip2p.replace(".", ""), base=2))
1389             self.str = ".".join([str(int(i, base=2)) for i in ip2p.split(".")])
1390         elif "0b" in str(ip2p):
1391             print("La ip", ip2p, "es bin")
1392             tmp = str(ip2p).replace("0b", "")
1393             n = 8
1394             self.bins = ".".join([tmp[i * n:i * n + n] for i, blah in enumerate(tmp[::n])])
1395             self.str = ".".join([str(int(tmp[i * n:i * n + n], base=2)) for i, blah in enumerate(tmp[::n])])
1396         else:
1397             raise
1398     else:
1399         print("Debug:", mode)
1400         raise NameError('No mode defined')
1401
1402 def update(self):
1403     ObjetoBase.update(self)
1404     self.image.set_tooltip_text(self.name + " (" + str(len(self.connections)) + "/" + str(self.max_connections)
1405                               + ")\n" + str(self.IP))
1406     submenu1 = self.builder.get_object("grid_rclick-sendpkg").get_submenu()
1407     print("Compcon: ", [x.name for x in self.compcon()])
1408
1409     for child in submenu1.get_children():
1410         if child.link.__class__.__name__ == "Switch" or child.link.__class__.__name__ == "Hub":
1411             child.hide()
1412         for con in self.compcon():
1413             if con.uuid not in [x.link.uuid for x in submenu1.get_children()]:
1414                 print("Not yet")
1415                 MeIt = Gtk.MenuItem.new_with_label(con.name)
1416                 MeIt.link = con
1417                 MeIt.connect("activate", self.send_pck)
1418                 submenu1.append(MeIt)
1419                 MeIt.show()
1420                 con.update()
1421             else:
1422                 print("\033[91m", con, "ya está en submenu1\033[0m")
1423                 pass
1424
1425     print("self.connections", self.connections)
1426
1427     if self.IP != None:
1428         objlst.update(self, "IP", str(self.IP))
1429
1430 #Ahora es cuando viene la parte de haber estudiado.
1431 #SÓLO ENVÍA PINGS, (ICMP)
1432 sub_N = 0
1433 def send_pck(self, *widget, to=None):

```

```

1433     global npack
1434     Sub_N = Computador.sub_N
1435     #nonlocal sub_N
1436     de = self
1437     print(widget)
1438     if to == None:
1439         to = widget[0].link
1440
1441     print("fnc send_pck from {} to {}".format(self.name, to.name))
1442
1443     if MainClase.has_ip(self) and MainClase.has_ip(to):
1444         print("Continuando")
1445     else:
1446         print("Un objeto no tiene IP")
1447         yonW = YesOrNoWindow("Uno o los dos objetos no tienen dirección IP", Yest="OK", Not="Ok también")
1448         yonR = yonW.run()
1449         yonW.destroy()
1450         raise Exception("Un objeto no tiene IP")
1451     #Ambos deben tener direccion ip
1452     #def __init__(self, header, payload, trailer, cabel=None):
1453     ping = Ping.create(0, self.IP, to.IP)
1454     Sub_N += 1
1455     npack += 1
1456
1457     print("PCK ICMP HEADER:", "{0:064b}".format(ping.icmp_header))
1458     print("PCK IPHEADER:", "{0:0160b}".format(ping.ip_header))
1459
1460     print("MAC's:", self.macdir, to.macdir)
1461     frame = eth(int(to.macdir), int(self.macdir), ping)
1462     frame.applytopack(ping)
1463     print("Pck frame:", ping.frame)
1464
1465     ping.animate(self, self.connections[0])
1466
1467     #Ver routing: https://en.wikipedia.org/wiki/IP_forwarding
1468     def packet_received(self, pck, *args, port=None):
1469         print("Hola, soy {} y he recibido un paquete, tal vez tenga que responder".format(self.name))
1470         #Si el tipo de ping es x, responder, si es y imprimir info
1471         if config.getboolean("DEBUG", "packet-received"):
1472             print(">Pck:", pck)
1473             if pck.frame != None:
1474                 frame="{0:0111b}".format(pck.frame)
1475                 print("\033[91m>>Atributos del paquete\033[00m")
1476                 totalen = pck.lenght + 14*8
1477                 print("Frame:", bin(pck.frame))
1478                 mac1 = "{0:0111b}".format(pck.frame)[0:6*8]
1479                 readmac = str(hex(int(mac1,2))).strip("0x")
1480                 print(">Mac1:", ":".join([readmac[i * 2:i * 2 + 2] for i,bl in enumerate(readmac[::2])]).upper())
1481                 readmac = str(hex(int( "{0:0111b}".format(pck.frame)[6*8+1:6*16+1] ,2))).strip("0x")
1482                 print(">Mac2:", ":".join([readmac[i * 2:i * 2 + 2] for i,bl in enumerate(readmac[::2])]).upper())
1483                 print("EtherType:", int(frame[12*8+1:8*14+1],2))
1484                 print("Resto==Bits:", int(frame[8*14+1::],2)==pck.bits)
1485                 print(pck.str)
1486
1487                 n, tmp = 8, pck.str[96:128]
1488                 print("IPs:", ":".join([str(int(tmp[i * n:i * n+n], base=2)) for i,blah in enumerate(tmp[::n])]))
1489                 tmp = pck.str[128:160]
1490                 print("IPd:", ":".join([str(int(tmp[i * n:i * n+n], base=2)) for i,blah in enumerate(tmp[::n])]))
1491
1492                 print("<<Fin de los atributos")
1493             n = 8
1494             tmp = pck.str[128:160]
1495             print(int(tmp,2), int(self.IP))
1496             if int(tmp,2) == int(self.IP):
1497                 ty = int("{0:064b}".format(pck.icmp_header)[8],2)
1498                 if ty == 8:
1499                     print("El paquete era para mí, voy a responder un gracias :D")
1500                     ping = Ping.create(1, self.IP, int(pck.str[96:128],2))
1501                     frame = eth(int("{0:0112b}".format(pck.frame)[6*8+1:6*16+1],2), int(self.macdir), ping)
1502                     frame.applytopack(ping)

```

```

1503
1504         ping.animate(self, self.connections[0])
1505     elif ty == 0:
1506         print("De nada")
1507     else:
1508         print("ty es:", ty)
1509
1510 class Servidor(Computador):
1511     def __init__(self, x, y, *args, name="Default", maxconnections=1, ip=None):
1512         self.objecttype = "Servidor"
1513
1514         push_elemento("Creado objeto {}".format(self.objecttype))
1515         self.img = resdir + "Server.*"
1516         ObjetoBase.__init__(self, x, y, self.objecttype, name=name)
1517         self.x = x
1518         self.y = y
1519         self.max_connections = maxconnections
1520         self.IP = self.ip()
1521
1522 #La clase para los objetos cable
1523 class Cable():
1524     def __init__(self, fromo, to, *color):
1525         lprint("Argumentos sobrantes: ", *color)
1526         self.objecttype = "Wire"
1527         self.fromobj = fromo
1528         self.toobj = to
1529         self.fromx = TheGrid.gridparser(fromo.x, TheGrid.wres,1)
1530         self.fromy = TheGrid.gridparser(fromo.y, TheGrid.hres,1)
1531         self.tox = TheGrid.gridparser(to.x, TheGrid.wres,1)
1532         self.toy = TheGrid.gridparser(to.y, TheGrid.hres,1)
1533         self.w = max(abs(fromo.realx - to.realx),3)
1534         self.h = max(abs(fromo.realy - to.realy),3)
1535
1536         self.cair()
1537
1538         self.x, self.y = min(fromo.x, to.x)-0.5, min(fromo.y, to.y)-0.5
1539
1540         TheGrid.moveto(self.image, self.x, self.y, layout=TheGrid.cables_lay)
1541         lprint("Puesto cable en: ", self.x, "; ", self.y)
1542
1543         self.image.show()
1544
1545         global cables
1546         cables.append(self)
1547         lprint("Todos los cables: ", cables)
1548
1549     def load(self):
1550         global cables
1551         self.cair()
1552         self.image.show()
1553         cables.append(self)
1554
1555         self.fromobj.connect(self.toobj, self)
1556
1557     def cair(self):
1558         fromo = self.fromobj
1559         to = self.toobj
1560         width, height = max(abs(self.fromobj.realx - self.toobj.realx),3), max(abs(self.fromobj.realy -
1561             ↪ self.toobj.realy),3)
1562         surface = cairo.ImageSurface(cairo.FORMAT_ARGB32, width, height)
1563         ctx = cairo.Context(surface)
1564
1565         #ctx.scale(width, height)
1566
1567         ctx.close_path ()
1568
1569         if config.getboolean("DEBUG", "show-cable-rectangle"):
1570             ctx.set_source_rgba(0, 0, 1, 0.1) # Solid color
1571             ctx.rectangle(0,0,width,height)
1572             ctx.fill()

```

```

1572
1573
1574     ctx.set_line_width(1.5)
1575     ctx.set_source_rgb(1,0,0)
1576     if (fromo.x < to.x and fromo.y < to.y) or (fromo.x > to.x and fromo.y > to.y):
1577         ctx.move_to(0, 0)
1578         ctx.line_to(width, height)
1579     elif fromo.x == to.x:
1580         ctx.move_to(width/2, 0)
1581         ctx.line_to(width/2, height)
1582     elif fromo.y == to.y:
1583         ctx.move_to(0, height/2)
1584         ctx.line_to(width, height/2)
1585     else:
1586         ctx.move_to(0, height)
1587         ctx.line_to(width, 0)
1588
1589     ctx.stroke()
1590
1591     self.image = gtk.Image.new_from_surface(surface)
1592     self.x, self.y = min(fromo.x, to.x)-0.5, min(fromo.y, to.y)-0.5
1593
1594     TheGrid.moveto(self.image, self.x, self.y, layout=TheGrid.cables_lay)
1595
1596     def delete(self):
1597         global cables
1598         cables.remove(self)
1599
1600         self.fromobj.cables.remove(self)
1601         self.toobj.cables.remove(self)
1602
1603         self.image.hide()
1604         print("\033[96mCable\033[00m", self, "\033[96mdeleted\033[00m")
1605         del self
1606
1607     save.classes = [ObjetoBase, Switch, Hub, Computador, Servidor, Cable]
1608
1609     #Función debug
1610     tmpvar = 0
1611     def theend():
1612         from random import randrange
1613         global tmpvar
1614         global TestC
1615         global TestD
1616
1617         scrolled = builder.get_object("scrolledwindow1")
1618         scrolled.get_vadjustment().set_value(0)
1619         scrolled.get_hadjustment().set_value(0)
1620
1621         if tmpvar>0:
1622             TestC.send_pck(to=TestD)
1623             tmpvar += 1
1624             if tmpvar > 4:
1625                 tmpvar = 1
1626
1627         else:
1628             TestC = Computador(2,3, name="From")
1629             TestC.IP = ip_address("192.168.1.38")
1630             #TestC.IP.set_str("192.168.1.38")
1631             print("{0:031b}".format(int(TestC.IP)))
1632
1633             TestD = Computador(8,3, name="To")
1634             #TestD.IP.set_str("192.168.1.42")
1635             TestD.IP = ip_address("192.168.1.42")
1636             print("{0:031b}".format(int(TestD.IP)))
1637
1638             bridge = Switch(4, 3, name="Bridge")
1639             bridge2= Switch(6, 3, name="Bridge2")
1640
1641             cable = Cable(TestC, bridge)

```

```

1642         cable2= Cable(bridge, bridge2)
1643         cable3= Cable(bridge2, TestD)
1644         TestC.connect(bridge, cable)
1645         bridge.connect(bridge2, cable2)
1646         TestD.connect(bridge2, cable3)
1647
1648         tmpvar += 1
1649
1650     #De momento sólo soportará el protocolo IPv4
1651     class packet():
1652         def __init__(self, header, trailer, payload, cabel=None):
1653             lprint("Creado paquete de res")
1654             self.header = header
1655             self.payload = payload
1656             self.trailer = trailer
1657             #self.packet = header + payload + trailer
1658
1659         def new_from_total(self, bits):
1660             print("Length (bits):", int(bin(bits)[18:33],2)*8)
1661             print("Real length:", int(len(bin(bits))-2 ))
1662             self.bits = bits
1663             self.length = int(bin(bits)[18:33],2)
1664             self.str = str("{0:0}" + str(int(int(bin(bits)[18:33],2)*8 )) + "b").format(self.bits)
1665             print(self.str)
1666
1667         def send(self, de):
1668             ##SIN TERMINAR##
1669             ##FALTA AÑADIR TODO LO DEL FRAME##
1670             if de.objecttype == "Computador":
1671                 to = de.connections[1]
1672                 self.animate(de, to)
1673
1674     #Composición de movimientos lineales en eje x e y
1675     #Siendo t=fps/s, v=px/s, v default = 84
1676     def animate(self, start, end, fps=120, v=200, color=None, port=None):
1677         if color == None:
1678             if self.color != None:
1679                 color = self.color
1680             else:
1681                 color = "#673AB7"
1682         from math import sqrt, pi
1683         #Long del cable
1684         try:
1685             cable = start.cables[[x.toobj for x in start.cables].index(end)]
1686         except ValueError:
1687             cable = start.cables[[x.fromobj for x in start.cables].index(end)]
1688         w, h = cable.w + TheGrid.sqres, cable.h + TheGrid.sqres
1689         x, y = cable.x*TheGrid.sqres-TheGrid.sqres/2, cable.y*TheGrid.sqres-TheGrid.sqres/2
1690         xi, yi = (start.x-0.5)*TheGrid.sqres-x, (start.y-0.5)*TheGrid.sqres-y
1691         xf, yf = end.x, end.y
1692         r = sqrt(cable.w**2+cable.h**2) #Píxeles totales
1693         t=r/v #Tiempo en segundos que durara la animacion
1694         tf = int(fps*t) #Fotogramas totales
1695         spf = 1/fps #Segundos por fotograma
1696
1697         sq = 12
1698         surface = cairo.ImageSurface(cairo.FORMAT_ARGB32, w, h)
1699         ctx = cairo.Context(surface)
1700         ctx.close_path()
1701         ctx.set_source_rgba(0,1,1,1)
1702         ctx.arc(-sq/2,-sq/2,sq/2,0,2*pi)
1703         ctx.fill()
1704         ctx.stroke()
1705         ctx.close_path()
1706
1707         image = gtk.Image.new_from_surface(surface)
1708         TheGrid.animat_lay.put(image,x,y)
1709         TheGrid.animat_lay.show_all()
1710
1711         #print("x: {}, y: {}, tf:{}, spf*m:{}, t: {}".format(x/TheGrid.sqres,y/TheGrid.sqres,tf,int(spf*1000), t))

```

```

1712     f = 0
1713     x,y = xi,yi
1714     sx,sy = (w-TheGrid.sqres)/tf,(h-TheGrid.sqres)/tf
1715     if start.x > end.x:
1716         sx = -sx
1717     if start.y > end.y:
1718         sy = -sy
1719
1720     def iteration():
1721         nonlocal f
1722         nonlocal x
1723         nonlocal y
1724         nonlocal ctx
1725         nonlocal surface
1726         nonlocal port
1727         if f <= tf:
1728             #Do things
1729             #print("Current f: {}; x,y: {}, {}".format(f, x,y))
1730             x += sx
1731             y += sy
1732
1733             del ctx
1734             surface = cairo.ImageSurface(cairo.FORMAT_ARGB32, w, h)
1735             ctx=cairo.Context(surface)
1736             ctx.set_source_rgba(*hex_to_rgba(color))
1737             ctx.arc(x,y,sq/2,0,2*pi)
1738             ctx.fill()
1739             image.set_from_surface(surface)
1740
1741             f += 1
1742             return True
1743         else:
1744             del ctx
1745             image.destroy()
1746             del surface
1747             #print("Paquete enviado a {}".format(end))
1748             if end.__class__.__name__ == "Switch":
1749                 for p in end.pall:
1750                     if end.pall[p].connection == start:
1751                         port = p
1752                         break
1753                 print("PORT:", port)
1754                 end.packet_received(self,port=port)
1755                 return False
1756             end.packet_received(self, port=port)
1757             return False
1758
1759     GObject.timeout_add(spf*1000, iteration)
1760
1761
1762     return True
1763
1764     def __str__(self):
1765         return "<" + str(packet) + ">"
1766
1767 # ETHERNET LAYER #
1768 #Usando DIX, más comun en IP
1769 #Al ser emulado no es necesario CRC Checksum
1770 #SIEMPRE 112 longitud (48*2+16)
1771 class eth(packet):
1772     #Se crea el header
1773     def __init__(self, destmac, sourcemac, *pack, EtherType=0x0800):
1774         def corrector(mac):
1775             if type(mac) == str:
1776                 mac2 = 0
1777                 for x in mac.split(":"):
1778                     mac2 = mac2 << 8 | int(x, 16)
1779             return mac2
1780         elif type(mac) == int:
1781             return mac

```

```

1782         else:
1783             raise Exception("MAC ERROR")
1784
1785         destmac = corrector(destmac)
1786         sourcemac = corrector(sourcemac)
1787         print("Destmac", "{0:048b}".format(destmac))
1788
1789         self.macheader = (destmac << (6*8+1) | sourcemac) << 16 | EtherType
1790         print(int("{0:0111b}".format(self.macheader)[0:6*8],2))
1791
1792     #Se le añade la payload al frame
1793     def applytopack(self, pack):
1794         self.pack = pack
1795         print(">Mach:", bin(self.macheader).replace("0b", ""))
1796         print(">Pck:", pack)
1797         print(pack.lenght)
1798         ret = (self.macheader << pack.lenght*8) | pack.bits
1799         pack.frame = ret
1800         pack.framesrt = None
1801         print("pack.len: {}, bits len: {}".format(pack.lenght*8, len(bin(pack.bits).strip("0b"))))
1802         print(">Ret:", bin(ret).replace("0b", ""))
1803         print(int("{0:0111b}".format(self.macheader)[0:6*8],2))
1804         return ret
1805
1806     def __str__(self):
1807         return str( bin(self.macheader) )
1808
1809 #Internet Layer
1810 class icmp(packet):
1811     def __init__(self, ipheader, icmpheader, payload):
1812         print("Len:", int(bin(ipheader)[18:33],2)-28)
1813         self.bits = (ipheader << 8*8 | icmpheader) << ( (int(bin(ipheader)[18:33],2) -28) * 8) | payload #BITS 16a31
1814         ↪ - 28
1815         packet.new_from_total(self, self.bits)
1816
1817     def __str__(self):
1818         return self.str
1819
1820 ### Application layer ###
1821
1822 #Estos paquetes pueden ser Request o Reply.
1823 #El header es de 20 bytes, la payload es de 8 + datos opcionales, pero el estándar es 64 bits.
1824 #Tipo de mensaje es 8 para request y 0 para reply. El ICMP es siempre 0.
1825 class Ping(icmp):
1826     identifi = 0
1827     def __init__(self):
1828         pass
1829
1830     def create(r, sourceip, desti_ip, *n, payload=int( 4.3*10**19 ) << 6 | 42, \
1831         flags=0b010, ttl=32):
1832         self = Ping()
1833         if r == 0:
1834             Type = 8
1835             self.color = "#4CAF50"
1836         if r == 1:
1837             Type = 0
1838             self.color = "#F44336"
1839
1840         self.payload = payload
1841
1842         vihlto = 0b0100010100000000
1843         #20 Ipheader + 8 ICMPHeader + Payload
1844         lenght = int( 20 + 8 + ( int(math.log(payload, 2))+1)/8 ) #In Bytes
1845         frag_off = 0b00000000000000
1846         protocol = 1
1847         checksum = 0 #No es necesario porque no hay cables
1848         sourceip = int(sourceip)
1849         desti_ip = int(desti_ip)
1850         identific = Ping.identifi

```



```

1851     Ping.identifi += 1
1852
1853     self.ip_header = (((((((((vihltos << 16 | lenght)<<16 | identific) << 3 | flags) << 13 | frag_off) \
1854 << 8 | ttl) << 8 | protocol) << 16 | checksum) << 32 | sourceip) << 32 | desti_ip)
1855
1856     identifier = 1*2**15 + 42 * 2**8 + 42
1857     Code = 0
1858     icmp_header_checksum = random.getrandbits(16)
1859     self.icmp_header = (((((((Type << 8) | Code)<< 16) | checksum) << 16) | identifier) << 16) | identific)
1860     self.pck = icmp(self.ip_header, self.icmp_header, self.payload)
1861
1862     self.str = self.pck.str
1863     self.lenght = self.pck.lenght
1864     self.bits = self.pck.bits
1865
1866     return self
1867
1868
1869
1870 #Ventana para configurar las variables de Config.ini
1871 #Nota: Por terminar
1872 class cfgWindow(MainClase):#MainClase):
1873     def __init__(self, *args):
1874         push_elemento("Invocada ventana de configuracion")
1875         writeonlog("Has invocado a la GRAN VENTANA DE CONFIGURACION <--- Boss")
1876         self.cfgventana = builder.get_object("cfgwindow")
1877         self.cfgventana.connect("key-press-event", self.on_key_press_event)
1878         self.cfgventana.connect("key-release-event", self.on_key_release_event)
1879         self.cfgventana.connect("delete-event", self.hidewindow)
1880
1881         builder.get_object("button2").connect("clicked", self.save)
1882
1883         self.eraselogs = builder.get_object("eraselogs")
1884         self.eraselogs.connect("clicked", self.borrarlogs)
1885
1886         self.cfgbtttn1 = builder.get_object("checkbutton1")
1887         self.cfgbtttn1.connect("toggled", self.bttntoggled)
1888         if config.getboolean("BOOLEANS", "print-key-pressed") == True:
1889             self.cfgbtttn1.set_active(True)
1890         else:
1891             self.cfgbtttn1.set_active(False)
1892
1893         booleans = {"print-key-pressed": "print-key-pressed"}
1894
1895         #TODO ESTO ES PARA LOS SPINNERS
1896
1897         #Todos los spinbuttons necesarios
1898         self.spinbuttons = [
1899             #[label, cfgsect, cfgkey, rangef, ranget, incrementf, increment],
1900             ["Win del wres", "GRAPHICS", "wres", 450, 1600, 5, 10],
1901             ["Win del hres", "GRAPHICS", "hres", 450, 1600, 5, 10],
1902             ["Wres del grid", "GRAPHICS", "viewport-wres", 20, 100, 1, 5],
1903             ["Hres del grid", "GRAPHICS", "viewport-hres", 15, 100, 1, 5],
1904             ["Res de los sq", "GRAPHICS", "viewport-sqres", 32, 128, 5, 10],
1905             ["Max logs", "DIRS", "Maxlogs", 3, 1000, 1, 5],
1906         ]
1907         self.createdspinbuttons = []
1908
1909         self.spinnergrid = builder.get_object("graph")
1910
1911         def forspin(spinner):
1912             spinbutton = Gtk.SpinButton.new(None, 0, 0)
1913             tmp1st = spinner
1914             label = Gtk.Label.new(tmp1st[0])
1915
1916             self.spinnergrid.insert_row(1)
1917
1918             #spinbutton.set_digits(0)
1919             spinbutton.set_numeric(True)
1920             spinbutton.set_range(tmp1st[3], tmp1st[4])

```

```

1921         spinbutton.set_increments(tmplst[5], tmplst[6])
1922         spinbutton.set_value(config.getfloat(tmplst[1], tmplst[2]))
1923
1924         #attach(child, left, top, width, height)
1925         self.spinnergrid.attach(label, 0, 1, 1, 1)
1926         self.spinnergrid.attach(spinbutton, 1, 1, 1, 1)
1927
1928         self.createdspinbuttons.append(spinbutton)
1929
1930     for spinner in self.spinbuttons:
1931         forspin(spinner)
1932
1933     #self.cfgventana.show_all()
1934
1935     def show(self, *args):
1936         self.cfgventana.show_all()
1937
1938     def on_key_press_event(self, widget, event):
1939         #global allkeys
1940         MainClase.on_key_press_event(self, widget, event)
1941         if "ESCAPE" in allkeys:
1942             push_elemento("Cerrada ventana de Configuracion")
1943             self.cfgventana.hide()
1944
1945         if ("CONTROL_L" in allkeys) and ("S" in allkeys):
1946             self.save()
1947             lprint(MainClase.on_key_press_event(self, widget, event))
1948
1949     def on_key_release_event(self, widget, event):
1950         MainClase.on_key_release_event(self, widget, event)
1951
1952     def btntoggled(self, *args):
1953         if self.cfgbtttn1.get_active() == True:
1954             push_elemento("print-key-pressed set True")
1955             config.set("BOOLEANS", "print-key-pressed", "True")
1956         if self.cfgbtttn1.get_active() == False:
1957             push_elemento("print-key-pressed set False")
1958             config.set("BOOLEANS", "print-key-pressed", "False")
1959
1960     def borrarlogs(self, *lala):
1961         #prompt = YesOrNoWindow("Seguro que quieres borrar los logs?")
1962         #if prompt.on_button_clicked(0) == True:
1963             push_elemento("Borrando logs")
1964             for the_file in os.listdir("logfiles/"):
1965                 file_path = os.path.join("logfiles/", the_file)
1966                 try:
1967                     if os.path.isfile(file_path):
1968                         os.unlink(file_path)
1969                 except e:
1970                     lprint(e)
1971
1972     def save(self, *args):
1973         #[label, cfgsect, cfgkey, rangef, ranget, incrementf, increment],
1974         lprint(self.createdspinbuttons)
1975         for i in range(len(self.createdspinbuttons)):
1976             tmplst = self.spinbuttons[i]
1977             config.set(tmplst[1], tmplst[2], int(self.createdspinbuttons[i].get_value()))
1978
1979         push_elemento("Configuracion guardada")
1980         with open(configdir, 'w') as cfgfile:
1981             lprint("Guardando archivo de configuracion")
1982             try:
1983                 config.write(cfgfile)
1984             except:
1985                 lprint("Error al guardar la configuracion")
1986
1987     def hidewindow(self, window, *event):
1988         window.hide()
1989         return True
1990

```

```

1991 class w_changethings(): #Oie tú, pedazo de subnormal, que cada objeto debe tener una...
1992     #0 tal vez no sea necesario... A la hora de llamar a la función, espera ¿Con quien estoy hablando?
1993     #Nota, ver notas escritas en la mesa
1994     def __init__(self, objeto):
1995         self.window = objeto.builder.get_object("changethings")
1996         self.name_entry = objeto.builder.get_object("changethings_name-entry")
1997         self.imagebutton = objeto.builder.get_object("changethings_imagebutton")
1998         self.applybutton = objeto.builder.get_object("chg_apply")
1999         self.applybutton.connect("clicked", self.apply)
2000         self.cancelbutton = objeto.builder.get_object("chg_cancel")
2001         self.cancelbutton.connect("clicked", self.cancel)
2002         self.window.connect("delete-event", self.hidewindow)
2003         self.window.connect("key-press-event", self.on_key_press_event)
2004         self.window.connect("key-release-event", self.on_key_release_event)
2005         objeto.builder.get_object("chg_MAC-regen").connect("clicked", self.regenclicked)
2006         print(objeto.builder.get_object("chg_MAC-regen").set_image(gtk.Image.new_from_stock("gtk-refresh", 1)))
2007
2008         self.link = objeto
2009         self.image = Gtk.Image.new_from_pixbuf(objeto.image.get_pixbuf())
2010
2011         #Esto es un quick fix que hace que las entry sólo acepten números
2012         def filter_numsdec(widget):
2013             text = widget.get_text().strip()
2014             widget.set_text(''.join([i for i in text if i in '0123456789']))
2015
2016         def filter_numshex(widget):
2017             text = widget.get_text().strip()
2018             widget.set_text(''.join([i for i in text if i in "0123456789ABCDEFabcdef"]))
2019
2020         for i in ["changethings_entry-IP" + str(x) for x in range(4)]:
2021             objeto.builder.get_object(i).connect("changed", filter_numsdec)
2022
2023         for i in ["chg_MAC-entry" + str(x) for x in range(0,5)]:
2024             objeto.builder.get_object(i).connect("changed", filter_numshex)
2025
2026         if objeto.objecttype != "Computer":
2027             objeto.builder.get_object("changethings_box-IP").destroy()
2028             objeto.builder.get_object("grid_label-IP").destroy()
2029
2030         #self.applybutton.connect("clicked", self.apply)
2031         #self.cancelbutton.connect("clicked", self.cancel)
2032
2033     def show(self, *widget):
2034         print("widget:", self.link)
2035         self.window.show_all()
2036         self.imagebutton.set_image(self.image)
2037         self.name_entry.set_text(self.link.name)
2038         tmplst = self.link.maddir.list()
2039         for i in tmplst:
2040             tmpentry = self.link.builder.get_object("chg_MAC-entry" + str(tmplst.index(i)))
2041             tmpentry.set_text(i)
2042
2043         #Hacer que muestre/oculte los campos de "IP"
2044         if self.link.objecttype == "Computer":
2045             try:
2046                 tmplst = str(self.link.IP).split(".")
2047                 print("TMPLST:", tmplst)
2048                 for i in tmplst:
2049                     tmpentry = self.link.builder.get_object("changethings_entry-IP" + str( tmplst.index(i) ))
2050                     tmpentry.set_text(i)
2051             except AttributeError: #Cuando no tiene una str definida
2052                 raise
2053             except TypeError:
2054                 raise
2055             except:
2056                 pass
2057             except:
2058                 raise
2059         else:
2060             pass

```

```

2061
2062 def apply(self, *npi):
2063     #acuerdate tambien de terminar esto
2064     #Nota: Hacer que compruebe nombres de una banlist, por ejemplo "TODOS"
2065     yonR = None
2066     lprint(npi)
2067
2068     self.link.name = self.name_entry.get_text()
2069     lprint([ self.link.builder.get_object(y).get_text() for y in ["chg_MAC-entry" + str(x) for x in range(0,6)]
2070             ↪ ])
2071     self.link.maddir.str = ":".join( [ self.link.builder.get_object(y).get_text() for y in ["chg_MAC-entry" +
2072             ↪ str(x) for x in range(6)] ])
2073     self.link.maddir.int = int(self.link.maddir.str.replace(":", ""))
2074     self.link.maddir.bin = "{0:048b}".format(self.link.maddir.int)
2075     if self.link.objectype == "Computer":
2076         try:
2077             self.link.IP = ip_address("." + self.link.builder.get_object(y).get_text() for y in
2078             ↪ ["changethings_entry-IP" + str(x) for x in range(4)] ])
2079         except ValueError:
2080             ip = ".".join( [ self.link.builder.get_object(y).get_text() for y in ["changethings_entry-IP" +
2081             ↪ str(x) for x in range(4)] ])
2082             if ip != "...":
2083                 print("No parece ser una IP válida:", ip)
2084                 yonW = YesOrNoWindow("{} no es una IP válida, por favor, introduzca una IP válida".format(ip),
2085                 ↪ Yest="OK", Not="Ok también")
2086                 yonR = yonW.run()
2087                 yonW.destroy()
2088         except:
2089             print(Exception)
2090             raise
2091
2092     lprint("self.link.name", self.link.name)
2093
2094     #self.link.image.set_tooltip_text(self.link.name + " (" + str(self.link.connections) + "/" +
2095     ↪ str(self.link.max_connections) + ")")
2096     self.link.update()
2097     self.window.hide()
2098     if yonR!=None:
2099         self.show()
2100
2101 def cancel(self, *npi):
2102     lprint(npi)
2103     self.window.hide()
2104
2105 def hidewindow(self, window, *event):
2106     window.hide()
2107     return True
2108
2109 def on_key_press_event(self, widget, event):
2110     #global allkeys
2111     MainClase.on_key_press_event(self, widget, event)
2112     if "ESCAPE" in allkeys:
2113         push_elemento("Cerrada ventana de Configuracion")
2114         self.window.hide()
2115
2116     if ("PERIOD" in allkeys) or ("KP_DECIMAL" in allkeys):
2117         widget.get_toplevel().child_focus(0)
2118
2119 def on_key_release_event(self, widget, event):
2120     MainClase.on_key_release_event(self, widget, event)
2121
2122 def regenclicked(self, widget):
2123     t = ObjetoBase.mac.genmac()[1].split(":")
2124     for i in t:
2125         tmpentry = self.link.builder.get_object("chg_MAC-entry" + str(t.index(i)))
2126         tmpentry.set_text(i)
2127         tmpentry.show()
2128
2129 class about(Gtk.AboutDialog):

```

```

2125     def __init__(self):
2126         self.win = builder.get_object("AboutWindow")
2127         self.win.connect("delete-event", self.destroy)
2128         self.win.connect("response", self.destroy)
2129         self.win.add_credit_section("Tutores", ["Julio Sánchez"])
2130         #self.win.add_credit_section("Contribuidores", [""])
2131         self = self.win
2132     def show(self, *args):
2133         print("Showing")
2134         self.win.show()
2135     def destroy(self, *args):
2136         self.win.hide()
2137         return True
2138
2139
2140 #Esta clase te permitirá deshacer acciones, algún día de un futuro lejano.
2141 class Undo():
2142     def __init__(self):
2143         self.lastactions = []
2144
2145 #Esta la pongo fuera porque lo mismo la necesito en otra clase
2146
2147 def exiting(self, *ahfjah):
2148     global log
2149     savelog()
2150     lprint("End time: " + time.strftime("%H:%M:%S"))
2151     print ("Window closed, exiting program")
2152     Gtk.main_quit()
2153
2154 def restart(*args):
2155     global log
2156     savelog()
2157     lprint("End time: " + time.strftime("%H:%M:%S"))
2158     lprint("Restarting program")
2159     print("\033[92m#####\033[00m")
2160     os.chdir(startcwd)
2161     os.execl(sys.executable, sys.executable, *sys.argv)
2162
2163 def returnTrue(*lala):
2164     return True
2165
2166 def nothing(self, *args):
2167     #Funcion Hugo
2168     pass
2169
2170 def leppard():
2171     lprint("Gunter lieben glauchen globen")
2172
2173 writeonlog("Esto ha llegado al final del codigo al parecer sin errores")
2174 writeonlog("0 tal vez no")
2175 MainClase()
2176
2177 #end()
2178
2179 lprint("Actual time: " + time.strftime("%H:%M:%S"))
2180 lprint("Complete load time: " + str(datetime.now() - startTime))
2181 push_elemento("Parece que esta cosa ha arrancado en tan solo " + str(datetime.now() - startTime))
2182 Gtk.main()
2183
2184 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             #Aquí 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 gi
4 import gi.repository
5 gi.require_version('Gtk', '3.0')
6 from gi.repository import Gtk, GObject, Gdk, GdkPixbuf
7
8 glade = "Interface2.glade"
9 last = 0
10 asgl = 1
11
12 ### AUN NO FUNCIONA ###
13
14 def save(allobjects, cabls, aslc=0):
15     global asgl
16     global last
17     if aslc | asgl:
18         asgl = 0

```

```

19         sw = loadWindow(mode=1)
20         fil = sw.run()
21         sw.destroy()
22     else:
23         fil = last
24     if fil != 0:
25         print(fil.split(".")[1])
26         if fil.split(".")[1] != "inv":
27             print("Nombre de archivo {} no tiene extensión .inv".format(fil))
28             fil += ".inv"
29         last = fil
30         try:
31             os.remove(fil)
32         except:
33             pass
34         print(allobjects)
35         with open(fil, "wb") as output:
36             pickle.dump((allobjects,cabls), output)
37
38 def load(allobjects, cabls):
39     lw = loadWindow()
40     fil = lw.run()
41     lw.destroy()
42     print(fil)
43     if fil != 0:
44         global last
45         global asgl
46         asgl = 0
47         last = fil
48         while len(allobjects) > 0:
49             allobjects[0].delete(pr=0)
50         while len(cabls) > 0:
51             cabls[0].delete()
52         with open(fil, "rb") as inpt:
53             allobj, cables = pickle.load(inpt)
54             print(allobj)
55             print(cables)
56             for obj in allobj:
57                 obj.load()
58             for cable in cables:
59                 cable.load()
60
61 class loadWindow(Gtk.Window):
62     def __init__(self, mode=0):
63         self.builder = Gtk.Builder()
64         self.builder.add_from_file(gladefile)
65         self.window = self.builder.get_object("window-filechooser_load")
66         filt = Gtk.FileFilter.new()
67         filt.add_pattern("*.inv")
68         filt.set_name("Archivos .inv")
69         self.window.add_filter(filt)
70         todos = Gtk.FileFilter.new()
71         todos.add_pattern("*")
72         todos.set_name("Todos los tipos de archivo")
73         self.window.add_filter(todos)
74         if mode == 1:
75             print("Saving")
76             self.window.set_action(Gtk.FileChooserAction.SAVE)
77             self.builder.get_object("window-filechooser_load-this").set_label("Guardar")
78
79     def run(self):
80         rs = self.window.run()
81         self.window.hide()
82         if rs == 1:
83             rs = self.window.get_filename()
84             self.window.destroy()
85         return rs
86     def destroy(self):
87         del self

```

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

