IES Palas Atenea

Proyecto de Investigación Bachillerato de excelencia

Programación, Redes y Código Libre

David Davó

Tutor
Julio Sánchez

27 de agosto de 2016

Índice general

Capítulo 1

Programación y código libre

Propuesta

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

1.1. Herramientas

El programa ha sido creado con herramientas de software libre. Según la Free Software Foundation "«Software libre» es el software que respeta la libertad de los usuarios y la comunidad. A grandes rasgos, significa que los usuarios tienen la libertad de ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software. Es decir, el «software libre» es una cuestión de libertad, no de precio. Para entender el concepto, piense en «libre» como en «libre expresión», no como en «barra libre». En inglés a veces decimos «libre software», en lugar de «free software», para mostrar que no queremos decir que es gratuito." –[3]

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

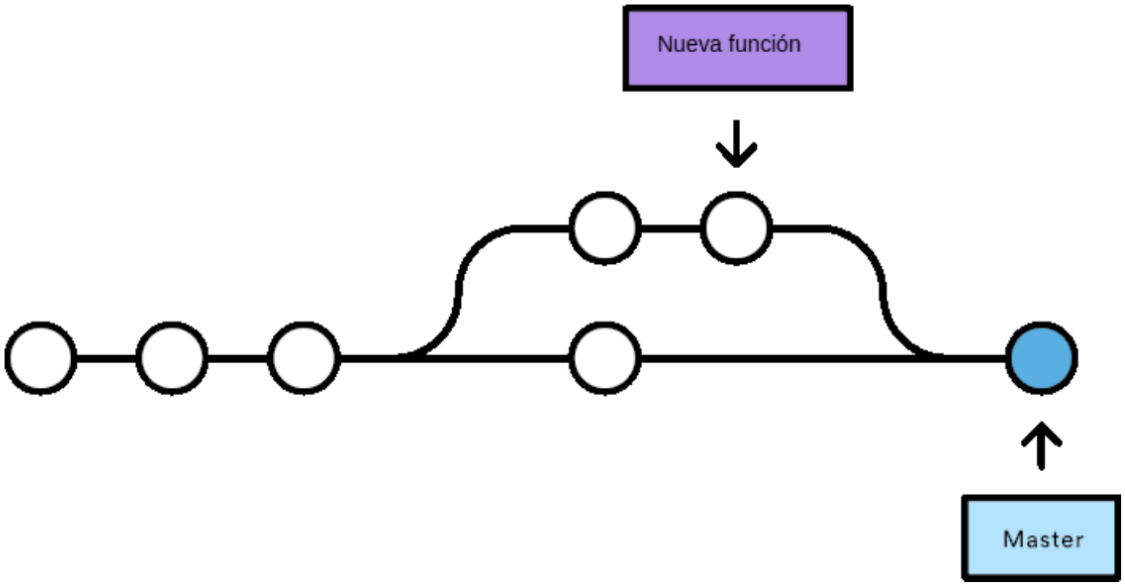

Figura 1.1: *Branching* con Git

GitHub es una plataforma de desarrollo colaborativo que te permite alojar tus repositorios Git. Su uso es gratuito si el código almacenado es público. Además, te permite tener, una wiki y una página web para tu proyecto, junto a otras funciones. Tanto el programa como este documento están disponibles en GitHub en el siguiente enlace. `https://github.com/daviddavo/InvProy`

1.1.3. LaTeX

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 aparencia 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 imperativo 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 mayoria de distros de Linux, por ejemplo en una instalación limpia de Ubuntu 16 (sin descargar paquetes adiccionales) el programa funciona perfectamente. Para usarlo en Python se ha tenido que importar la libreria 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.

Capítulo 2

Redes Informáticas

Historia

Internet, tal y como lo conocemos ahora, haciendo uso de IPv6, HTML5, CSS3 no existe hasta hace recientemente, 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 *Open Systems Interconnection* (Interconexión de Sistemas Abiertos) (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 en sí, sino son entidades de distintas normativas internacionales.

Capa	PDU	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 *frames* 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 continúo 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 carácteres.	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.

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 linear.

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 çable 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 podia 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*

Capítulo 3

El simulador de redes

3.1. Instalación

PONER INFORMACIÓN SOBRE LA INSTALACION CON UN ADMINISTRADOR DE PAQUETES

3.1.1. 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:

```
cd Descargas
git clone https://github.com/daviddavo/InvProy.git
```

Una vez ya tenemos el repositorio de git clonado:

```
cd InvProy
python3 Main.py
```

Glosario y acrónimos

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

Bit ***Bi****nary digi****t****, o dígito binario. Cada dígito del sistema de numeración binario.*

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 fucniones 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)

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." [2][Topología]

Topología de red Configuración espacial o física de la red. (Ver 2.3 pág.5)

Bibliografía

[1] BICSI. *Network Design Basics for Cabling Professionals*. 2002.

[2] Real Academia Española. *Diccionario de la lengua española, ed. XXIII*. 2014.

[3] FSF. *Filosofía del Proyecto GNU*. 2013. url: https://www.gnu.org/philosophy/philosophy.html.

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

Índice de figuras

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), carácteres/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

```
# -*- coding: utf-8 -*-
#!/usr/bin/env python3

'''
    InvProy - Simulador de Redes / Proyecto de Investigación
    https://github.com/daviddavo/InvProy
    Copyright (C) 2016  David Davó Laviña  david@ddavo.me  http://ddavo.me

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.

    ////////////////////////////

    Este programa es código libre: Puedes redistribuirlo y/o modificarlo
    bajo los términos de la licencia GNU General Public License tal y como
    publicado por la Free Software Foundation, ya sea la versión 3 de layout
    licencia o la más reciente.

    Este programa es distribuido con la esperanza de que sea útil, pero
    SIN NINGUNA GARANTÍA; sin siquiera la garantía implícita de COMERCIABILIDAD
    o de la APTITUD DE LA MISMA PARA UN PROPÓSITO PARTICULAR. Ver la GNU General
    Public License para más detalles.

    Debes haber recibido una copia de la GNU General Public License con
    este programa, si no es así, ver <http://www.gnu.org/licenses/>.
'''
from datetime import datetime
startTime = datetime.now()

import configparser, os, csv, sys, time, random, math
import xml.etree.ElementTree as xmltree
from ipaddress import ip_address
from random import choice

#Esto hace que el programa se pueda ejecutar fuera de la carpeta.
startcwd = os.getcwd()

```

```
48  try:
49      os.chdir(os.path.dirname(sys.argv[0]))
50  except:
51      pass
52
53  os.system("clear")
54  print("\033[91m##############################\033[00m")
55
56  print("InvProy  Copyright (C) 2016  David Davó Laviña\ndavid@ddavo.me   <http://ddavo.me>\n\
57  This program comes with ABSOLUTELY NO WARRANTY; for details go to 'Ayuda > Acerca de'\n\
58  This is free software, and you are welcome to redistribute it\n\
59  under certain conditions\n")
60
61  try: #Intenta importar los modulos necesarios
62      #sys.path.append("Modules/")
63      import Modules.Test
64  except:
65      print("Error: No se han podido importar los modulos...")
66      sys.exit()
67
68  #Aqui importamos los modulos del programa que necesitamos...
69
70  from Modules.logmod import *
71  from Modules import save
72
73  def lprint(*objects, sep=" ", end="\n", file=sys.stdout, flush=False):
74      print(*objects, sep=sep, end=end, file=file, flush=flush)
75      thing=str()
76      for i in objects:
77          thing += str(i) + sep
78      writeonlog(thing)
79
80  lprint("Start loading time: " + time.strftime("%H:%M:%S"))
81
82  try:
83      #Importando las dependencias de la interfaz
84      import gi
85      gi.require_version('Gtk', '3.0')
86      from gi.repository import Gtk, GObject, Gdk, GdkPixbuf
87  except:
88      lprint("Por favor, instala PyGObject en tu ordenador. \n  En ubuntu suele ser 'apt-get install python3-gi'\n  En
          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+
              "+str(Gtk.get_major_version())+"."+str(Gtk.get_minor_version())+"."+str(Gtk.get_micro_version()))
176     except Exception as e:
177         lprint("Error: No se ha podido cargar la interfaz.")
178         if "required" in str(e):
179             xmlroot = xmltree.parse(gladefile).getroot()
180             lprint("Necesario Gtk+ "+ xmlroot[0].attrib["version"]+".0", end="\n")
181             lprint(">Estas usando
                  Gtk+"+str(Gtk.get_major_version())+"."+str(Gtk.get_minor_version())+"."+str(Gtk.get_micro_version()))
182         else:
183             lprint("Debug:", e)
184         sys.exit()
```

```
185
186 #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 appendtoclicked(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 ESTÄ, QUE AÑADA ATRIBUTOS A LA LISTA.
286             it1 = self.tree.append(None, row=[obj.name, obj.objectype])
287             it2 = self.tree.append(it1, row=["MAC", str(obj.macdir)])
288             itc = self.tree.append(it1, row=["Conexiones", "{}/{}".format(len(obj.connections),
                    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.objectype])
307             self.tree.set_value(obj.trdic["Connections"], 1, "{}/{}".format(len(obj.connections),
                    obj.max_connections))
308             for i in obj.connections:
309                 print(i.__repr__(), obj.trcondic)
310                 if i in obj.trcondic.keys():
311                     self.tree.set_value(obj.trcondic[i], 0, i.name)
312                 else:
313                     r = self.tree.append(obj.trdic["Connections"], row=[i.name, ""])
314                     obj.trcondic[i] = r
315 
316         def show(self, *args):
317             rev = self.revealer.get_reveal_child()
318             if rev:
319                 self.panpos = builder.get_object("paned1").get_position()
```

```
320
321             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 deberiamos 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 
398             #Para no tener que hacer click continuamente
399             if ("Q" in allkeys):
400                 self.toolbutton_clicked(builder.get_object("toolbutton3"))
401             if "W" in allkeys:
402                 self.toolbutton_clicked(builder.get_object("toolbutton4"))
403             if "E" in allkeys:
404                 self.toolbutton_clicked(builder.get_object("toolbutton5"))
405             if "R" in allkeys:
406                 self.toolbutton_clicked(builder.get_object("toolbutton6"))
407             if "T" in allkeys:
408                 self.toolbutton_clicked(builder.get_object("toolbutton7"))
409             return keyname
410 
411         #Al dejar de pulsar la tecla deshace lo anterior.
412         def on_key_release_event(self, widget, event):
413             keynameb = Gdk.keyval_name(event.keyval).upper()
414             if config.getboolean("BOOLEANS", "print-key-pressed") == True:
415                 lprint("Key %s (%d) released" % (keynameb, event.keyval))
416             global allkeys
417             allkeys.discard(keynameb)
418 
419         def drag_drop(widget, context, x, y, time):
420             push_elemento( "Drag drop at " + str(x) +"," + str(y) )
421 
422         #Comprueba si el objeto tiene una ip asignada
423         def has_ip(self):
424             try:
425                 if self.IP != None:
426                     return True
427                 else:
428                     return False
429             except:
430                 return False
431 
432         def save(*args):
433             global cables
434             global allobjects
435             lscl = 0
436             try:
437                 if args[1].get_label() == "gtk-save-as":
438                     print("Guardando como")
439                     lscl = 1
440             except:
441                 pass
442             save.save(allobjects,cables, aslc=lscl)
443             push_elemento("Guardando...")
444         def load(*args):
445             global cables
446             global allobjects
447             save.load(allobjects,cables)
448             push_elemento("Cargando...")
449         def new(*args):
450             global allobjects
451             global cables
452             save.last = 0
453             while len(allobjects) > 0:
454                 allobjects[0].delete(pr=0)
455             while len(cables) > 0:
456                 cables[0].delete()
457 
458         def new(*args):
459             global cables
```

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

```
530         ### INICIO CAIRO
531 
532         width, height, sq = self.wres*self.sqres, self.hres*self.sqres, self.sqres
533         surface = cairo.ImageSurface(cairo.FORMAT_ARGB32, width, height)
534         ctx = cairo.Context(surface)
535         ctx.close_path ()
536         ctx.set_source_rgba(0,0,0,1)
537         ctx.set_line_width(1)
538 
539         for i in range(self.wres):
540             ctx.move_to(i*sq, 0)
541             ctx.line_to(i*sq, height)
542         for i in range(self.hres):
543             ctx.move_to(0, i*sq)
544             ctx.line_to(width, i*sq)
545 
546 
547         ctx.stroke()
548         self.image = Gtk.Image.new_from_surface(surface)
549         ### FINAL DE LO DE CAIRO
550 
551         self.backgr_lay.put(self.image, 0, 0)
552 
553         def subshow(widget):
554             #Para que no aparezca arriba a la izquierda:
555             scrolled = builder.get_object("scrolledwindow1")
556             scrolled.get_vadjustment().set_value(height/3)
557             scrolled.get_hadjustment().set_value(width/3)
558 
559         if config.getboolean("GRAPHICS","start-centered"):
560             builder.get_object("window1").connect("show", subshow)
561         self.overlay.show_all()
562         self.contadorback = 0
563 
564     def moveto(self, image, x, y, *args, layout=None):
565         if x < self.wres and y < self.hres:
566             if layout == None:
567                 layout = self.mainport
568             elif str(layout.__class__.__name__) == "Layout":
569                 layout = layout
570             else:
571                 print("layout.__class__.__name__", layout.__class__.__name__)
572             if image in layout.get_children():
573                 layout.move(image, x*self.sqres, y*self.sqres)
574             else:
575                 layout.put(image, x*self.sqres, y*self.sqres)
576         else:
577             print("\033[31mError: Las coordenadas se salen del grid\033[00m")
578 
579     def clicked_on_grid(self, widget, event, *args):
580         global clicked
581         global bttnclicked
582         global allobjects
583         global areweputtingcable
584         self.contadorback += 1
585 
586         push_elemento("Clicked on grid @" + str(self.gridparser(event.x, self.wres)) + "," +
              str(self.gridparser(event.y, self.hres)))
587 
588         if self.searchforobject(self.gridparser(event.x, self.wres), self.gridparser(event.y, self.hres)) == False:
589             if clicked == 1:
590                 push_elemento("Clicked: " + str(clicked) + " bttnclicked: " + str(bttnclicked))
591                 if bttnclicked == "Router":
592                     Router(self.gridparser(event.x, self.wres), self.gridparser(event.y, self.hres))
593                     push_elemento("Creado objeto router")
594                 elif bttnclicked == "toolbutton4":
595                     Switch(self.gridparser(event.x, self.wres), self.gridparser(event.y, self.hres))
596                     push_elemento("Creado objeto switch")
597                 elif bttnclicked == "toolbutton6":
598                     Computador(self.gridparser(event.x, self.wres), self.gridparser(event.y, self.hres))
```

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

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

```
737             self.image.show()
738
739             self.macdir = self.mac()
740             print("MAC:", self.macdir, int(self.macdir), bin(self.macdir))
741             if ip == None:
742                 print("No ip definida")
743                 self.ipstr = "None"
744
745             #Ahora vamos con lo de aparecer en la lista de la izquierda,
746             #aunque en realidad es un grid
747             lista = objlst
748             self.trlst = lista.append(self)
749             self.image.set_tooltip_text(self.name + " (" + str(len(self.connections)) + "/" + str(self.max_connections)
                  + ")\n" + self.ipstr)
750
751             self.window_changethings = w_changethings(self)
752             self.builder.get_object("grid_rclick-name").connect("activate", self.window_changethings.show)
753
754             self.cnt = 0 #Se me olvido que hace esta cosa
755
756         def load(self):
757             global cnt_objects
758             global cnt_rows
759             global allobjects
760             self.builder = Gtk.Builder()
761             self.builder.add_from_file(gladefile)
762             self.menuemergente = self.builder.get_object("grid_rclick")
763             self.builder.get_object("grid_rclick-disconnect_all").connect("activate", self.disconnect)
764             self.builder.get_object("grid_rclick-delete").connect("activate", self.delete)
765             self.builder.get_object("grid_rclick-debug").connect("activate", self.debug)
766             self.connections = []
767             self.cables = []
768             cnt_objects += 1
769             self.__class__.cnt += 1
770             allobjects.append(self)
771             self.image = gtk.Image.new_from_file(self.imgdir)
772             self.resizetogrid(self.image)
773             TheGrid.moveto(self.image, self.x-1, self.y-1)
774             self.image.show()
775             lista = builder.get_object("grid2")
776             lista.insert_row(cnt_rows)
777             self.label = Gtk.Label.new(self.name)
778             lista.attach(self.label, 0, cnt_rows, 1, 1)
779             cnt_rows += 1
780             self.label.show()
781             self.image.set_tooltip_text(self.name + " (" + str(len(self.connections)) + "/" + str(self.max_connections)
                  + ")\n" + self.ipstr)
782             self.window_changethings = w_changethings(self)
783             self.builder.get_object("grid_rclick-name").connect("activate", self.window_changethings.show)
784
785             print("CABLES",self.cables)
786
787         #Esta funcion retorna una str cuando se usa el objeto. En lugar de <0xXXXXXXXX object>
788         def __str__(self):
789             return  "<Tipo: " + self.objectype +" | Name: " + self.name + " | Pos: " + str(self.x) + ", " + str(self.y) +
                  ">"
790
791         def debug(self, *args):
792             print("DEBUG")
793             print("MAC:", self.macdir, int(self.macdir))
794
795         def rclick(self, event):
796             global rclick_Object
797             rclick_Object = self
798
799             print(self)
800             lprint("rclick en", self.x, self.y, self.objectype, "\nConnections: ", end="")
801             lprint(self.connections)
802             self.rmenu = self.menuemergente
803             if self.objectype == "Computer" and len(self.compcon()) > 0:
```

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

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

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

```
1013
1014                if de in [x.link for x in iterc]:
1015                    print("\033[91mSelf in\033[00m", self)
1016
1017                for cable in self.cables:
1018                    if cable.fromobj == self or cable.toobj == self:
1019                        cable.delete()
1020                        break
1021
1022                de.update()
1023
1024                if self.__class__.__name__ == "Switch":
1025                    self.disconnectport(de)
1026                elif de.__class__.__name__ == "Switch":
1027                    de.disconnectport(self)
1028
1029            self.update()
1030
1031        def delete(self, *widget, conf=1, pr=1):
1032            if pr == 1:
1033                yonW = YesOrNoWindow("¿Estás seguro de que quieres eliminar " + self.name + " definitivamente? El objeto
                      será imposible de recuperar y te hechará de menos.")
1034                yonR = yonW.run()
1035                yonW.destroy()
1036            else:
1037                yonR = 1
1038            if yonR == 1:
1039                self.disconnect(0, de="All")
1040                objlst.delete(self)
1041                self.image.destroy()
1042                global allobjects
1043                allobjects.remove(self)
1044            elif yonR == 0:
1045                print("Piénsatelo dos veces")
1046            else:
1047                raise
1048
1049        def packet_received(self, pck, *args, port=None):
1050            print("Hola, soy {} y he recibido un paquete, pero no sé que hacer con él".format(self.name))
1051            if config.getboolean("DEBUG", "packet-received"):
1052                print(">Pck:",pck)
1053                if pck.frame != None:
1054                    print("\033[91m>>Atributos del paquete\033[00m")
1055                    totalen = pck.lenght + 14*8
1056                    wfr = bformat(pck.frame, (totalen+14)*8)
1057                    print(">Wfr:",wfr)
1058                    mac1 = "{0:0111b}".format(pck.frame)[0:6*8]
1059                    print(">Mac:", int(mac1,2))
1060                    readmac = str(hex(int(mac1,2))).strip("0x")
1061                    print(":".join([readmac[i * 2:i * 2 + 2] for i,bl in enumerate(readmac[::2])]).upper())
1062
1063                    print("<<Fin de los atributos")
1064
1065    npack = 0
1066
1067    class Router(ObjetoBase):
1068        cnt = 1
1069        def __init__(self, x, y, *args, name="Default"):
1070            global cnt_objects
1071            self.objectype = "Router"
1072            push_elemento("Creado Objeto Router")
1073
1074            ObjetoBase.__init__(self, x, y, self.objectype, name=name)
1075            self.x = x
1076            self.y = y
1077
1078        def __del__(self, *args):
1079            push_elemento("Eliminado objeto")
1080            del self
1081
```

```
1082  class Switch(ObjetoBase):
1083      cnt = 1
1084      #El objeto puerto
1085      class Port():
1086          def __init__(self, switch):
1087              self.id = switch.portid
1088              self.dic = switch.pdic
1089              self.all = switch.pall
1090              switch.portid += 1
1091              self.switch = switch
1092              self.connection = None
1093              self.all[self.id] = self
1094              self.dic[self.id] = self.connection
1095          def connect(self, connection):
1096              self.connection = connection
1097              self.dic[self.id] = self.connection
1098          def disconnect(self):
1099              self.connection = None
1100              self.dic[self.id] = self.connection
1101          def is_available(self):
1102              if self.connection == None:
1103                  return True
1104              return False
1105
1106      class w_switch_table(Gtk.ApplicationWindow):
1107          def __init__(self, switch):
1108              self.link = switch
1109              builder = switch.builder
1110              builder.get_object("window_switch-table_button").connect("clicked", self.hide)
1111              builder.get_object("window_switch-table").connect("delete-event", self.hide)
1112              self.store = Gtk.ListStore(str,int,int,int)
1113
1114              self.view = builder.get_object("window_switch-table-TreeView")
1115              self.view.set_model(self.store)
1116              for i, column_title in enumerate(["MAC", "Puerto", "TTL (s)"]):
1117                  renderer = Gtk.CellRendererText()
1118                  column = Gtk.TreeViewColumn(column_title, renderer, text=i)
1119                  column.set_sort_column_id(i)
1120                  self.view.append_column(column)
1121              self.ticking = False
1122              builder.get_object("window_switch-table").set_keep_above(True)
1123
1124          def show(self, *a):
1125              self.ticking = True
1126              GObject.timeout_add(1001, self.tick)
1127              for row in self.store:
1128                  row[2] = row[3] - time.time()
1129              self.link.builder.get_object("window_switch-table").show_all()
1130
1131          def hide(self, window, *event):
1132              self.link.builder.get_object("window_switch-table").hide()
1133              self.ticking = False
1134              return True
1135          def append(self, lst):
1136              lst.append(lst[2])
1137              for row in self.store:
1138                  row[2] = row[3] - time.time()
1139              print(lst)
1140              row = self.store.append(lst)
1141              print(self.view.get_property("visible"))
1142              if self.view.get_property("visible") == True:
1143                  self.ticking = True
1144                  GObject.timeout_add(1001, self.tick)
1145
1146          def tick(self):
1147              for row in self.store:
1148                  row[2] = row[3] - time.time()
1149                  if row[2] <= 0:
1150                      try:
1151                          self.store.remove(row.iter)
```

```
1152                        self.link.table.remove(row)
1153                    except:
1154                        pass
1155            if len(self.store) == 0:
1156                self.ticking = False
1157            return self.ticking
1158        def remove(self, lst):
1159            for row in self.store:
1160                if row == lst:
1161                    self.store.remove(row.iter)
1162                    self.link.table
1163                    break
1164            pass
1165
1166    def __init__(self, x, y, *args, name="Default", maxconnections=5):
1167        self.objectype = "Switch"
1168        self.portid = 0
1169        self.pdic = {}
1170        self.pall = {}
1171
1172        push_elemento("Creado objeto Switch")
1173        self.imgdir = resdir + "Switch.*"
1174        ObjetoBase.__init__(self, x, y, self.objectype, name=name, maxconnections=maxconnections)
1175        self.x = x
1176        self.y = y
1177        self.timeout = 20 #Segundos
1178
1179        for p in range(self.max_connections):
1180            self.Port(self)
1181        print(self.pall)
1182
1183        self.table = [
1184        #[MAC, port, expiration]
1185        ]
1186        self.wtable = self.w_switch_table(self)
1187        child = Gtk.MenuItem.new_with_label("Routing Table")
1188        self.builder.get_object("grid_rclick").append(child)
1189        child.connect("activate", self.wtable.show)
1190        child.show()
1191
1192        self.ch = child
1193
1194    def load(self):
1195        ObjetoBase.load(self)
1196        del self.wtable
1197        self.table = []
1198        self.wtable = self.w_switch_table(self)
1199
1200        del self.ch
1201        child = Gtk.MenuItem.new_with_label("Routing Table")
1202        self.builder.get_object("grid_rclick").append(child)
1203        child.connect("activate", self.wtable.show)
1204        child.show()
1205
1206        self.ch = child
1207
1208
1209    def connectport(self, objeto):
1210        for port in self.pall:
1211            if self.pall[port].is_available():
1212                self.pall[port].connect(objeto)
1213                break
1214        print(self.pdic)
1215
1216    def disconnectport(self, objeto):
1217        for p in self.pdic:
1218            print("i: {}, idx: {}".format(p,self.pdic[p]))
1219            if objeto == self.pdic[p]:
1220                self.pall[p].disconnect()
1221                break
```

```
1222            print(self.pdic)
1223
1224        def packet_received(self, pck, port=None):
1225            macd = "{0:0112b}".format(pck.frame)[0:6*8]
1226            macs = "{0:0112b}".format(pck.frame)[6*8+1:6*16+1]
1227
1228            #LO PRIMERO: AÑADIRLO A LA TABLA
1229            readmac = str(hex(int(macs,2))).upper().replace("0X", "")
1230            readmac = ":".join([readmac[i * 2:i * 2 + 2] for i,bl in enumerate(readmac[::2])])
1231
1232            for tab in self.table:
1233                if tab[2] <= time.time():
1234                    print("Ha llegado tu hora")
1235                    self.table.remove(tab)
1236                    self.wtable.remove(tab)
1237                if tab[0] == int(macd,2):
1238                    print("TAB[0] == mcd")
1239                    tab[2] = int(time.time()+self.timeout)
1240                    for row in self.wtable.store:
1241                        print(row[0], tab[0])
1242                        if int(row[0].replace(":",""),16) == tab[0]:
1243                            row[3] = int(time.time()+self.timeout)
1244            if int(macs,2) not in [x[0] for x in self.table]:
1245                tmp = [int(macs,2), port, int(time.time()+self.timeout)]
1246                self.table.append(tmp)
1247                tmp = [readmac, port, int(time.time()+self.timeout)]
1248                self.wtable.append(tmp)
1249
1250            ################################
1251
1252            #ObjetoBase.packet_received(self, pck)
1253
1254            ttl  = int(pck.str[64:72],2)
1255            ttlnew = "{0:08b}".format(ttl-1)
1256            pck.str = "".join(( pck.str[:64], ttlnew, pck.str[72:] ))
1257
1258            print("self.macdir",int(self.macdir), int("{0:0112b}".format(pck.frame)[6*8+1:6*16+1],2))
1259            print("TTL:", int(pck.str[64:72],2), pck.str[64:72])
1260
1261            print("Soy {} y mi deber es entregar el paquete a {}".format(self.name,int(macd,2)))
1262            print("El paquete llegó por el puerto  {}".format(port))
1263            dic = {}
1264            for i in self.connections:
1265                dic[int(i.macdir)] = i
1266            print("Connections MAC's:", dic)
1267
1268            #Cambiamos los bits de macs
1269            #Si macd en conn, enviarle el paquete
1270            #Si existe una tabla de enrutamiento que contiene una ruta para macd, enviar por ahi
1271            #Si no, enviar al siguiente, y así
1272            print(">MAAAC:",int(macd,2), "DIIIC:")
1273            if int(macd,2) in dic and ttl > 0:
1274                pck.animate(self, dic[int(macd,2)])
1275
1276            elif int(macd,2) in [x[0] for x in self.table]:
1277                for x in self.table:
1278                    if x[0] == int(macd,2):
1279                        pck.animate(self, self.pdic[x[1]])
1280
1281            elif "Switch" in [x.objectype for x in self.connections] and ttl >= 0:
1282                print("Ahora lo enviamos al siguiente router")
1283                print(int(macd,2), dic)
1284                tmplst = self.connections[:] #Crea una nueva copia de la lista
1285                print(tmplst)
1286                for i in tmplst:
1287                    if int(macs,2) == int(i.macdir):
1288                        print("REMOVING", i)
1289                        tmplst.remove(i)
1290                try:
1291                    tmplst.remove(*[x for x in tmplst if x.objectype == "Computer"])
```

```
1292                 except TypeError:
1293                     pass
1294                 print("Tmplst:", tmplst)
1295                 obj = choice(tmplst)
1296                 print("Sending to:", obj)
1297                 pck.animate(self, obj)
1298 
1299     def debug(self, *args):
1300         print(self.pdic)
1301         print("MyMac:", self.macdir)
1302         row_format ="{:>20}" * 3
1303         print(row_format.format("MAC", "NXT", "EXP s"))
1304         for row in self.table:
1305             if row[1] == None:
1306                 row[1] = "None"
1307             if int(row[2]-time.time()) <= 0:
1308                 self.table.remove(row)
1309             print(row_format.format(row[0], row[1], int(row[2]-int(time.time()))))
1310 
1311 #¿Tengo permisos de escritura?, no se si tendré permisos
1312 #Update: Si los tenía
1313 class Hub(ObjetoBase):
1314     cnt = 1
1315     def __init__(self, x, y, *args, name="Default", maxconnections=4, ip=None):
1316         self.objectype = "Hub"
1317         push_elemento("Creado objeto Hub")
1318         self.imgdir = resdir + "Hub.*"
1319         ObjetoBase.__init__(self, x, y, self.objectype, name=name)
1320         self.x = x
1321         self.y = y
1322 
1323     def packet_received(self,pck,port=None):
1324         ttl  = int(pck.str[64:72],2)
1325         macs = "{0:0112b}".format(pck.frame)[6*8+1:6*16+1]
1326         ttlnew = "{0:08b}".format(ttl-1)
1327         pck.str = "".join(( pck.str[:64], ttlnew, pck.str[72:] ))
1328         if ttl >= 0:
1329             for obj in self.connections:
1330                 pck.animate(self, obj)
1331 
1332 class Computador(ObjetoBase):
1333     cnt = 1
1334     def __init__(self, x, y, *args, name="Default", maxconnections=1, ip=None):
1335         self.objectype = "Computer"
1336 
1337         push_elemento("Creado objeto Computador")
1338         self.img = resdir + "Comp.*"
1339         ObjetoBase.__init__(self, x, y, self.objectype, name=name)
1340         self.x = x
1341         self.y = y
1342         self.max_connections = maxconnections
1343         self.IP = None
1344         self.update()
1345 
1346     class ip():
1347         def __init__(self, *args, ipstr="None"):
1348             self.str = ipstr
1349 
1350         def __str__(self):
1351             return self.str
1352 
1353         def set_str(self, str):
1354             self.str = str
1355             self.parser(str, 0)
1356 
1357         def set_bin(self, binar):
1358             t = binar
1359             print(bin(t))
1360             if "0b" not in str(t) and "." in str(t):
1361                 print("Type is str")
```

```
1362                self.bins = t
1363            elif "0b" in str(bin(t)) and "." not in str(bin(t)):
1364                print("Type is binar")
1365                self.bin = t
1366            else:
1367                print("Error:", t)
1368            self.parser(t, 1)
1369
1370        #ip2p stands 4 'ip to parse'
1371        def parser(self, ip2p, mode):
1372            #mode 0: str2b
1373            if mode == 0:
1374                tmplst = ip2p.split(".")
1375                toreturn = []
1376                for i in tmplst:
1377                    i = int(i)
1378                    toreturn.append("{0:08b}".format(i))
1379                self.bins = ".".join(toreturn)
1380                self.bin = int(self.bins.replace(".", ""), base=2)
1381                return self.bins
1382
1383            #mode 1: b2str
1384            elif mode == 1:
1385                if "0b" not in str(ip2p):
1386                    self.bin = bin(int(ip2p.replace(".", ""), base=2))
1387                    self.str = ".".join([str(int(i, base=2)) for i in ip2p.split(".")])
1388                elif "0b" in str(ip2p):
1389                    print("La ip", ip2p, "es bin")
1390                    tmp = str(ip2p).replace("0b", "")
1391                    n = 8
1392                    self.bins = ".".join([tmp[i * n:i * n+n] for i,blah in enumerate(tmp[::n])])
1393                    self.str = ".".join([str(int(tmp[i * n:i * n+n], base=2)) for i,blah in enumerate(tmp[::n])])
1394                else:
1395                    raise
1396            else:
1397                print("Debug:", mode)
1398                raise NameError('No mode defined')
1399
1400    def update(self):
1401        ObjetoBase.update(self)
1402        self.image.set_tooltip_text(self.name + " (" + str(len(self.connections)) + "/" + str(self.max_connections)
              + ")\n" + str(self.IP))
1403        submenu1 = self.builder.get_object("grid_rclick-sendpkg").get_submenu()
1404        print("Compcon: ", [x.name for x in self.compcon()])
1405
1406        for child in submenu1.get_children():
1407            if child.link.__class__.__name__ == "Switch" or child.link.__class__.__name__ == "Hub":
1408                child.hide()
1409                for con in self.compcon():
1410                    if con.uuid not in [x.link.uuid for x in submenu1.get_children()]:
1411                        print("Not yet")
1412                        MeIt = Gtk.MenuItem.new_with_label(con.name)
1413                        MeIt.link = con
1414                        MeIt.connect("activate", self.send_pck)
1415                        submenu1.append(MeIt)
1416                        MeIt.show()
1417                        con.update()
1418                    else:
1419                        print("\033[91m",con, "ya está en submenu1\033[0m")
1420                        pass
1421
1422                print("self.connections", self.connections)
1423
1424        if self.IP != None:
1425            objlst.update(self,"IP", str(self.IP))
1426
1427    #Ahora es cuando viene la parte de haber estudiado.
1428    #SÓLO ENVÍA PINGS, (ICMP)
1429    sub_N = 0
1430    def send_pck(self, *widget, to=None):
```

```
1431            global npack
1432            Sub_N = Computador.sub_N
1433            #nonlocal sub_N
1434            de = self
1435            print(widget)
1436            if to == None:
1437                to = widget[0].link
1438
1439            print("fnc send_pck from {} to {}".format(self.name, to.name))
1440
1441            if MainClase.has_ip(self) and MainClase.has_ip(to):
1442                print("Continuando")
1443            else:
1444                print("Un objeto no tiene IP")
1445                yonW = YesOrNoWindow("Uno o los dos objetos no tienen dirección IP", Yest="OK", Not="Ok también")
1446                yonR = yonW.run()
1447                yonW.destroy()
1448                raise Exception("Un objeto no tiene IP")
1449            #Ambos deben tener direccion ip
1450            #def __init__(self, header, payload, trailer, cabel=None):
1451            ping = Ping.create(0, self.IP, to.IP)
1452            Sub_N += 1
1453            npack += 1
1454
1455            print("PCK ICMP HEADER:", "{0:064b}".format(ping.icmp_header))
1456            print("PCK IPHEADER:", "{0:0160b}".format(ping.ip_header))
1457
1458            print("MAC's:", self.macdir, to.macdir)
1459            frame = eth(int(to.macdir), int(self.macdir), ping)
1460            frame.applytopack(ping)
1461            print("Pck frame:", ping.frame)
1462
1463            ping.animate(self, self.connections[0])
1464
1465        #Ver routing: https://en.wikipedia.org/wiki/IP_forwarding
1466        def packet_received(self, pck, *args, port=None):
1467            print("Hola, soy {} y he recibido un paquete, tal vez tenga que responder".format(self.name))
1468            #Si el tipo de ping es x, responder, si es y imprimir info
1469            if config.getboolean("DEBUG", "packet-received"):
1470                print(">Pck:",pck)
1471                if pck.frame != None:
1472                    frame="{0:0111b}".format(pck.frame)
1473                    print("\033[91m>>Atributos del paquete\033[00m")
1474                    totalen = pck.lenght + 14*8
1475                    print("Frame:", bin(pck.frame))
1476                    mac1 = "{0:0111b}".format(pck.frame)[0:6*8]
1477                    readmac = str(hex(int(mac1,2))).strip("0x")
1478                    print(">Mac1:", ":".join([readmac[i * 2:i * 2 + 2] for i,bl in enumerate(readmac[::2])]).upper())
1479                    readmac = str(hex(int( "{0:011b}".format(pck.frame)[6*8+1:6*16+1] ,2))).strip("0x")
1480                    print(">Mac2:", ":".join([readmac[i * 2:i * 2 + 2] for i,bl in enumerate(readmac[::2])]).upper())
1481                    print("EtherType:", int(frame[12*8+1:8*14+1],2))
1482                    print("Resto==Bits:", int(frame[8*14+1::],2)==pck.bits)
1483                    print(pck.str)
1484
1485                    n, tmp = 8, pck.str[96:128]
1486                    print("IPs:", ".".join([str(int(tmp[i * n:i * n+n], base=2)) for i,blah in enumerate(tmp[::n])]))
1487                    tmp = pck.str[128:160]
1488                    print("IPd:", ".".join([str(int(tmp[i * n:i * n+n], base=2)) for i,blah in enumerate(tmp[::n])]))
1489
1490                    print("<<Fin de los atributos")
1491            n = 8
1492            tmp = pck.str[128:160]
1493            print(int(tmp,2), int(self.IP))
1494            if int(tmp,2) == int(self.IP):
1495                ty = int("{0:064b}".format(pck.icmp_header)[:8],2)
1496                if ty == 8:
1497                    print("El paquete era para mí, voy a responder un gracias :D")
1498                    ping = Ping.create(1, self.IP, int(pck.str[96:128],2))
1499                    frame = eth(int("{0:0112b}".format(pck.frame)[6*8+1:6*16+1],2), int(self.macdir), ping)
1500                    frame.applytopack(ping)
```

```
1501
1502                ping.animate(self, self.connections[0])
1503            elif ty == 0:
1504                print("De nada")
1505            else:
1506                print("ty es:", ty)
1507
1508 class Servidor(Computador):
1509     def __init__(self, x, y, *args, name="Default", maxconnections=1, ip=None):
1510         self.objectype = "Servidor"
1511
1512         push_elemento("Creado objeto {}".format(self.objectype))
1513         self.img = resdir + "Server.*"
1514         ObjetoBase.__init__(self, x, y, self.objectype, name=name)
1515         self.x = x
1516         self.y = y
1517         self.max_connections = maxconnections
1518         self.IP = self.ip()
1519
1520 #La clase para los objetos cable
1521 class Cable():
1522     def __init__(self, fromo, to, *color):
1523         lprint("Argumentos sobrantes: ", *color)
1524         self.objectype = "Wire"
1525         self.fromobj = fromo
1526         self.toobj = to
1527         self.fromx = TheGrid.gridparser(fromo.x, TheGrid.wres,1)
1528         self.fromy = TheGrid.gridparser(fromo.y, TheGrid.hres,1)
1529         self.tox = TheGrid.gridparser(to.x, TheGrid.wres,1)
1530         self.toy = TheGrid.gridparser(to.y, TheGrid.hres,1)
1531         self.w   = max(abs(fromo.realx - to.realx),3)
1532         self.h   = max(abs(fromo.realy - to.realy),3)
1533
1534         self.cair()
1535
1536         self.x, self.y = min(fromo.x, to.x)-0.5, min(fromo.y, to.y)-0.5
1537
1538         TheGrid.moveto(self.image, self.x, self.y, layout=TheGrid.cables_lay)
1539         lprint("Puesto cable en: ", self.x, "; ", self.y)
1540
1541         self.image.show()
1542
1543         global cables
1544         cables.append(self)
1545         lprint("Todos los cables: ", cables)
1546
1547     def load(self):
1548         global cables
1549         self.cair()
1550         self.image.show()
1551         cables.append(self)
1552
1553         self.fromobj.connect(self.toobj, self)
1554
1555     def cair(self):
1556         fromo = self.fromobj
1557         to    = self.toobj
1558         width, height = max(abs(self.fromobj.realx - self.toobj.realx),3), max(abs(self.fromobj.realy -
              self.toobj.realy),3)
1559         surface = cairo.ImageSurface(cairo.FORMAT_ARGB32, width, height)
1560         ctx = cairo.Context(surface)
1561
1562         #ctx.scale(width, height)
1563
1564         ctx.close_path ()
1565
1566         if config.getboolean("DEBUG", "show-cable-rectangle"):
1567             ctx.set_source_rgba(0, 0, 1, 0.1) # Solid color
1568             ctx.rectangle(0,0,width,height)
1569             ctx.fill()
```

```
1570
1571
1572            ctx.set_line_width(1.5)
1573            ctx.set_source_rgb(1,0,0)
1574            if (fromo.x < to.x and fromo.y < to.y) or (fromo.x > to.x and fromo.y > to.y):
1575                ctx.move_to(0, 0)
1576                ctx.line_to(width, height)
1577            elif fromo.x == to.x:
1578                ctx.move_to(width/2, 0)
1579                ctx.line_to(width/2, height)
1580            elif fromo.y == to.y:
1581                ctx.move_to(0, height/2)
1582                ctx.line_to(width, height/2)
1583            else:
1584                ctx.move_to(0, height)
1585                ctx.line_to(width, 0)
1586
1587            ctx.stroke()
1588
1589            self.image = gtk.Image.new_from_surface(surface)
1590            self.x, self.y = min(fromo.x, to.x)-0.5, min(fromo.y, to.y)-0.5
1591
1592            TheGrid.moveto(self.image, self.x, self.y, layout=TheGrid.cables_lay)
1593
1594        def delete(self):
1595            global cables
1596            cables.remove(self)
1597
1598            self.fromobj.cables.remove(self)
1599            self.toobj.cables.remove(self)
1600
1601            self.image.hide()
1602            print("\033[96mCable\033[00m", self, "\033[96mdeleted\033[00m")
1603            del self
1604
1605    save.classes = [ObjetoBase, Switch, Hub, Computador, Servidor, Cable]
1606
1607    #De momento sólo soportará el protocolo IPv4
1608    class packet():
1609        def __init__(self, header, trailer, payload, cabel=None):
1610            lprint("Creado paquete de res")
1611            self.header = header
1612            self.payload = payload
1613            self.trailer = trailer
1614            #self.packet = header + payload + trailer
1615
1616        def new_from_total(self, bits):
1617            print("Length (bits):", int(bin(bits)[18:33],2)*8)
1618            print("Real length:", int(len(bin(bits))-2 ))
1619            self.bits = bits
1620            self.lenght = int(bin(bits)[18:33],2)
1621            self.str = str("{0:0"+str(int(int(bin(bits)[18:33],2)*8 ))+"b}").format(self.bits)
1622            print(self.str)
1623
1624        def send(self, de):
1625            ##SIN TERMINAR##
1626            ##FALTA AÑADIR TODO LO DEL FRAME##
1627            if de.objectype == "Computador":
1628                to = de.connections[1]
1629            self.animate(de, to)
1630
1631        #Composición de movimientos lineales en eje x e y
1632        #Siendo t=fps/s, v=px/s, v default = 84
1633        def animate(self, start, end, fps=120, v=200, color=None, port=None):
1634            if color == None:
1635                if self.color != None:
1636                    color = self.color
1637                else:
1638                    color = "#673AB7"
1639            from math import sqrt, pi
```

```
1640            #Long del cable
1641            try:
1642                cable = start.cables[[x.toobj for x in start.cables].index(end)]
1643            except ValueError:
1644                cable = start.cables[[x.fromobj for x in start.cables].index(end)]
1645            w, h = cable.w + TheGrid.sqres, cable.h + TheGrid.sqres
1646            x, y = cable.x*TheGrid.sqres-TheGrid.sqres/2, cable.y*TheGrid.sqres-TheGrid.sqres/2
1647            xi, yi = (start.x-0.5)*TheGrid.sqres-x, (start.y-0.5)*TheGrid.sqres-y
1648            xf, yf = end.x, end.y
1649            r = sqrt(cable.w**2+cable.h**2) #Pixeles totales
1650            t=r/v #Tiempo en segundos que durara la animacion
1651            tf = int(fps*t) #Fotogramas totales
1652            spf = 1/fps #Segundos por fotograma
1653
1654            sq = 12
1655            surface = cairo.ImageSurface(cairo.FORMAT_ARGB32, w, h)
1656            ctx = cairo.Context(surface)
1657            ctx.close_path()
1658            ctx.set_source_rgba(0,1,1,1)
1659            ctx.arc(-sq/2,-sq/2,sq/2,0,2*pi)
1660            ctx.fill()
1661            ctx.stroke()
1662            ctx.close_path()
1663
1664            image = gtk.Image.new_from_surface(surface)
1665            TheGrid.animat_lay.put(image,x,y)
1666            TheGrid.animat_lay.show_all()
1667
1668            #print("x: {}, y: {}, tf:{}, spf*m:{}, t: {}".format(x/TheGrid.sqres,y/TheGrid.sqres,tf,int(spf*1000), t))
1669            f = 0
1670            x,y = xi,yi
1671            sx,sy = (w-TheGrid.sqres)/tf,(h-TheGrid.sqres)/tf
1672            if start.x > end.x:
1673                sx = -sx
1674            if start.y > end.y:
1675                sy = -sy
1676
1677            def iteration():
1678                nonlocal f
1679                nonlocal x
1680                nonlocal y
1681                nonlocal ctx
1682                nonlocal surface
1683                nonlocal port
1684                if f <= tf:
1685                    #Do things
1686                    #print("Current f: {}; x,y: {}, {}".format(f, x,y))
1687                    x += sx
1688                    y += sy
1689
1690                    del ctx
1691                    surface = cairo.ImageSurface(cairo.FORMAT_ARGB32, w, h)
1692                    ctx=cairo.Context(surface)
1693                    ctx.set_source_rgba(*hex_to_rgba(color))
1694                    ctx.arc(x,y,sq/2,0,2*pi)
1695                    ctx.fill()
1696                    image.set_from_surface(surface)
1697
1698                    f += 1
1699                    return True
1700                else:
1701                    del ctx
1702                    image.destroy()
1703                    del surface
1704                    #print("Paquete enviado a {}".format(end))
1705                    if end.__class__.__name__ == "Switch":
1706                        for p in end.pall:
1707                            if end.pall[p].connection == start:
1708                                port = p
1709                                break
```

```
1710                    print("PORT:", port)
1711                    end.packet_received(self,port=port)
1712                    return False
1713                end.packet_received(self, port=port)
1714                return False
1715
1716        GObject.timeout_add(spf*1000, iteration)
1717
1718
1719        return True
1720
1721    def __str__(self):
1722        return "<" + str(packet) + ">"
1723
1724 # ETHERNET LAYER #
1725 #Usando DIX, más comun en IP
1726 #Al ser emulado no es necesario CRC Checksum
1727 #SIEMPRE 112 longitud (48*2+16)
1728 class eth(packet):
1729     #Se crea el header
1730     def __init__(self, destmac, sourcemac, *pack, EtherType=0x0800):
1731         def corrector(mac):
1732             if type(mac) == str:
1733                 mac2 = 0
1734                 for x in mac.split(":"):
1735                     mac2 = mac2 << 8 | int(x, 16)
1736                 return mac2
1737             elif type(mac) == int:
1738                 return mac
1739             else:
1740                 raise Exception("MAC ERROR")
1741
1742         destmac = corrector(destmac)
1743         sourcemac = corrector(sourcemac)
1744         print("Destmac", "{0:048b}".format(destmac))
1745
1746         self.macheader = (destmac << (6*8+1) | sourcemac) << 16 | EtherType
1747         print(int("{0:0111b}".format(self.macheader)[0:6*8],2))
1748
1749     #Se le añade la payload al frame
1750     def applytopack(self, pack):
1751         self.pack = pack
1752         print(">Mach:", bin(self.macheader).replace("0b", ""))
1753         print(">Pck:", pack)
1754         print(pack.lenght)
1755         ret = (self.macheader << pack.lenght*8) | pack.bits
1756         pack.frame = ret
1757         pack.framesrt = None
1758         print("pack.len: {}, bits len: {}".format(pack.lenght*8, len(bin(pack.bits).strip("0b"))))
1759         print(">Ret:", bin(ret).replace("0b",""))
1760         print(int("{0:0111b}".format(self.macheader)[0:6*8],2))
1761         return ret
1762
1763     def __str__(self):
1764         return str( bin(self.macheader) )
1765
1766 #Internet Layer
1767 class icmp(packet):
1768     def __init__(self, ipheader, icmpheader, payload):
1769         print("Len:", int(bin(ipheader)[18:33],2)-28)
1770         self.bits = (ipheader << 8*8 | icmpheader) << ( (int(bin(ipheader)[18:33],2) -28) * 8) | payload #BITS 16a31
              □   - 28
1771         packet.new_from_total(self, self.bits)
1772
1773     def __str__(self):
1774         return self.str
1775
1776
1777 ### Application layer ###
1778
```

```
1779  #Estos paquetes pueden ser Request o Reply.
1780  #El header es de 20 bytes, la payload es de 8 + datos opcionales, pero el estándar es 64 bits.
1781  #Tipo de mensaje es 8 para request y 0 para reply. El ICMP es siempre 0.
1782  class Ping(icmp):
1783      identifi = 0
1784      def __init__(self):
1785          pass
1786
1787      def create(r, sourceip, desti_ip, *n, payload=int( 4.3*10**19 ) << 6 | 42, \
1788          flags=0b010, ttl=32):
1789          self = Ping()
1790          if r == 0:
1791              Type = 8
1792              self.color = "#4CAF50"
1793          if r == 1:
1794              Type = 0
1795              self.color = "#F44336"
1796
1797          self.payload = payload
1798
1799          vihltos   = 0b0100010100000000
1800          #20 Ipheader + 8 ICMPHEader + Payload
1801          lenght    = int( 20 + 8 + ( int(math.log(payload, 2))+1)/8 ) #In Bytes
1802          frag_off  = 0b0000000000000
1803          protocol  = 1
1804          checksum  = 0 #No es necesario porque no hay cables
1805          sourceip  = int(sourceip)
1806          desti_ip  = int(desti_ip)
1807          identific = Ping.identifi
1808          Ping.identifi += 1
1809
1810          self.ip_header = (((((((((vihltos << 16 | lenght)<<16 | identific) << 3 | flags) << 13 | frag_off) \
1811          << 8 | ttl) << 8 | protocol) << 16 | checksum) << 32 | sourceip) << 32 | desti_ip)
1812
1813          identifier = 1*2**15 + 42 * 2**8 + 42
1814          Code = 0
1815          icmp_header_checksum = random.getrandbits(16)
1816          self.icmp_header = (((((((Type << 8) | Code)<< 16) | checksum) << 16) | identifier) << 16) | identific)
1817          self.pck = icmp(self.ip_header, self.icmp_header, self.payload)
1818
1819          self.str = self.pck.str
1820          self.lenght = self.pck.lenght
1821          self.bits = self.pck.bits
1822
1823          return self
1824
1825
1826
1827  #Ventana para configurar las variables de Config.ini
1828  #Nota: Por terminar
1829  class cfgWindow(MainClase):#MainClase):
1830      def __init__(self, *args):
1831          push_elemento("Invocada ventana de configuracion")
1832          writeonlog("Has invocado a la GRAN VENTANA DE CONFIGURACION <--- Boss")
1833          self.window = builder.get_object("cfgWindow")
1834
1835          #Todos los spinbuttons necesarios
1836          self.spinbuttons = [
1837              #[label, cfgsect, cfgkey, rangef, ranget, incrementf, increment, parent],
1838              #["Anchura de la ventana", "GRAPHICS", "wres", 450, 1600, 5, 10],
1839              #["Altura de la ventana", "GRAPHICS", "hres", 450, 1600, 5, 10],
1840              ["Anchura", "GRAPHICS", "viewport-wres", 20, 100, 1, 5, builder.get_object("cfgwnd-frame-grid")],
1841              ["Altura", "GRAPHICS", "viewport-hres", 15, 100, 1, 5, builder.get_object("cfgwnd-frame-grid")],
1842              ["Lado de los cuadros", "GRAPHICS", "viewport-sqres", 32, 128, 5, 10,
                    builder.get_object("cfgwnd-frame-grid")],
1843              #["Max logs", "DIRS", "Maxlogs", 3, 1000, 1, 5, builder.get_object("cfgwnd-frame-grid")],
1844          ]
1845          self.createdspinbuttons = []
1846          lprint("spinbuttons: " + str(len(self.spinbuttons)))
1847          def set_spinner(lst):
```

```
1848                #spinbutton = builder.get_object(spinner)
1849                spinbutton = Gtk.SpinButton.new(None, 0, 0)
1850                tmplst = lst
1851                label = Gtk.Label.new(tmplst[0])
1852
1853                tmplst[7].insert_row(1)
1854
1855                #spinbutton.set_digits(0)
1856                spinbutton.set_numeric(True)
1857                spinbutton.set_range(tmplst[3], tmplst[4])
1858                spinbutton.set_increments(tmplst[5], tmplst[6])
1859                spinbutton.set_value(config.getfloat(tmplst[1], tmplst[2]))
1860                spinbutton.set_margin_left(5)
1861
1862                #attach(child, left, top, width, height)
1863                tmplst[7].attach(label, 0, 1, 1, 1)
1864                tmplst[7].attach(spinbutton, 1, 1, 1, 1)
1865
1866                self.createdspinbuttons.append(spinbutton)
1867
1868            for spinner in self.spinbuttons:
1869                set_spinner(spinner)
1870
1871        def show(self, *args):
1872            self.window.show_all()
1873
1874        def on_key_press_event(self, widget, event):
1875            #global allkeys
1876            MainClase.on_key_press_event(self,widget,event)
1877            if "ESCAPE" in allkeys:
1878                push_elemento("Cerrada ventana de Configuracion")
1879                self.cfgventana.hide()
1880
1881            if ("CONTROL_L" in allkeys) and ("S" in allkeys):
1882                self.save()
1883            lprint(MainClase.on_key_press_event(self,widget,event))
1884
1885        def on_key_release_event(self, widget, event):
1886            MainClase.on_key_release_event(self, widget, event)
1887
1888        def bttntoggled(self, *args):
1889            if self.cfgbttn1.get_active() == True:
1890                push_elemento("print-key-pressed set True")
1891                config.set("BOOLEANS", "print-key-pressed", "True")
1892            if self.cfgbttn1.get_active() == False:
1893                push_elemento("print-key-pressed set False")
1894                config.set("BOOLEANS", "print-key-pressed", "False")
1895
1896        def borrarlogs(self, *lala):
1897            #prompt = YesOrNoWindow("Seguro que quieres borrar los logs?")
1898            #if prompt.on_button_clicked(0) == True:
1899            push_elemento("Borrando logs")
1900            for the_file in os.listdir("logfiles/"):
1901                file_path = os.path.join("logfiles/", the_file)
1902                try:
1903                    if os.path.isfile(file_path):
1904                        os.unlink(file_path)
1905                except e:
1906                    lprint(e)
1907
1908        def save(self, *args):
1909            #[label, cfgsect, cfgkey, rangef, ranget, incrementf, increment],
1910            lprint(self.createdspinbuttons)
1911            for i in range(len(self.createdspinbuttons)):
1912                tmplst = self.spinbuttons[i]
1913                config.set(tmplst[1], tmplst[2], int(self.createdspinbuttons[i].get_value()))
1914
1915            push_elemento("Configuracion guardada")
1916            with open(configdir, 'w') as cfgfile:
1917                lprint("Guardando archivo de configuracion")
```

```
1918                try:
1919                    config.write(cfgfile)
1920                except:
1921                    lprint("Error al guardar la configuracion")
1922
1923        def hidewindow(self, window, *event):
1924            window.hide()
1925            return True
1926
1927    class w_changethings(): #Oie tú, pedazo de subnormal, que cada objeto debe tener una...
1928        #O tal vez no sea necesario... A la hora de llamar a la función, espera ¿Con quien estoy hablando?
1929        #Nota, ver notas escritas en la mesa
1930        def __init__(self, objeto):
1931            self.window = objeto.builder.get_object("changethings")
1932            self.name_entry = objeto.builder.get_object("changethings_name-entry")
1933            self.imagebutton = objeto.builder.get_object("changethings_imagebutton")
1934            self.applybutton = objeto.builder.get_object("chg_apply")
1935            self.applybutton.connect("clicked", self.apply)
1936            self.cancelbutton = objeto.builder.get_object("chg_cancel")
1937            self.cancelbutton.connect("clicked", self.cancel)
1938            self.window.connect("delete-event", self.hidewindow)
1939            self.window.connect("key-press-event", self.on_key_press_event)
1940            self.window.connect("key-release-event", self.on_key_release_event)
1941            objeto.builder.get_object("chg_MAC-regen").connect("clicked", self.regenclicked)
1942            print(objeto.builder.get_object("chg_MAC-regen").set_image(gtk.Image.new_from_stock("gtk-refresh", 1)))
1943
1944            self.link = objeto
1945            self.image = Gtk.Image.new_from_pixbuf(objeto.image.get_pixbuf())
1946
1947            #Esto es un quick fix que hace que las entry sólo acepten números
1948            def filter_numsdec(widget):
1949                text = widget.get_text().strip()
1950                widget.set_text(''.join([i for i in text if i in '0123456789']))
1951
1952            def filter_numshex(widget):
1953                text = widget.get_text().strip()
1954                widget.set_text("".join([i for i in text if i in "0123456789ABCDEFabcdef"]))
1955
1956            for i in ["changethings_entry-IP" + str(x) for x in range(4)]:
1957                objeto.builder.get_object(i).connect("changed", filter_numsdec)
1958
1959            for i in ["chg_MAC-entry" + str(x) for x in range(0,5)]:
1960                objeto.builder.get_object(i).connect("changed", filter_numshex)
1961
1962            if objeto.objectype != "Computer":
1963                objeto.builder.get_object("changethings_box-IP").destroy()
1964                objeto.builder.get_object("grid_label-IP").destroy()
1965
1966            #self.applybutton.connect("clicked", self.apply)
1967            #self.cancelbutton.connect("clicked", self.cancel)
1968
1969        def show(self, *widget):
1970            print("widget:", self.link)
1971            self.window.show_all()
1972            self.imagebutton.set_image(self.image)
1973            self.name_entry.set_text(self.link.name)
1974            tmplst = self.link.macdir.list()
1975            for i in tmplst:
1976                tmpentry = self.link.builder.get_object("chg_MAC-entry" + str(tmplst.index(i)))
1977                tmpentry.set_text(i)
1978
1979            #Hacer que muestre/oculte los campos de "IP"
1980            if self.link.objectype == "Computer":
1981                try:
1982                    tmplst = str(self.link.IP).split(".")
1983                    print("TMPLST:", tmplst)
1984                    for i in tmplst:
1985                        tmpentry = self.link.builder.get_object("changethings_entry-IP" + str( tmplst.index(i) ))
1986                        tmpentry.set_text(i)
1987                except AttributeError: #Cuando no tiene una str definida
```

```
1988                raise
1989                pass
1990            except TypeError:
1991                raise
1992                pass
1993            except:
1994                raise
1995        else:
1996            pass
1997
1998    def apply(self, *npi):
1999        #acuerdate tambien de terminar esto
2000        #Nota: Hacer que compruebe nombres de una banlist, por ejemplo "TODOS"
2001        yonR = None
2002        lprint(npi)
2003
2004        self.link.name = self.name_entry.get_text()
2005        lprint([ self.link.builder.get_object(y).get_text() for y in ["chg_MAC-entry" + str(x) for x in range(0,6)]
                ])
2006        self.link.macdir.str = ":".join( [ self.link.builder.get_object(y).get_text() for y in ["chg_MAC-entry" +
                str(x) for x in range(6)] ])
2007        self.link.macdir.int = int(self.link.macdir.str.replace(":",""), 16)
2008        self.link.macdir.bin = "{0:048b}".format(self.link.macdir.int)
2009        if self.link.objectype == "Computer":
2010            try:
2011                self.link.IP = ip_address(".".join( [ self.link.builder.get_object(y).get_text() for y in
                        ["changethings_entry-IP" + str(x) for x in range(4)] ]))
2012            except ValueError:
2013                ip = ".".join( [ self.link.builder.get_object(y).get_text() for y in ["changethings_entry-IP" +
                        str(x) for x in range(4)] ])
2014                if ip != "...":
2015                    print("No parece ser una IP válida:", ip)
2016                    yonW = YesOrNoWindow("{} no es una IP válida, por favor, introduzca una IP válida".format(ip),
                            Yest="OK", Not="Ok también")
2017                    yonR = yonW.run()
2018                    yonW.destroy()
2019            except:
2020                print(Exception)
2021                raise
2022
2023        lprint("self.link.name", self.link.name)
2024
2025        #self.link.image.set_tooltip_text(self.link.name + " (" + str(self.link.connections) + "/" +
                str(self.link.max_connections) + ")")
2026        self.link.update()
2027        self.window.hide()
2028        if yonR!=None:
2029            self.show()
2030
2031    def cancel(self, *npi):
2032        lprint(npi)
2033        self.window.hide()
2034
2035    def hidewindow(self, window, *event):
2036        window.hide()
2037        return True
2038
2039    def on_key_press_event(self, widget, event):
2040        #global allkeys
2041        MainClase.on_key_press_event(self,widget,event)
2042        if "ESCAPE" in allkeys:
2043            push_elemento("Cerrada ventana de Configuracion")
2044            self.window.hide()
2045
2046        if ("PERIOD" in allkeys) or ("KP_DECIMAL" in allkeys):
2047            widget.get_toplevel().child_focus(0)
2048
2049
2050    def on_key_release_event(self, widget, event):
2051        MainClase.on_key_release_event(self, widget, event)
```

```
2052
2053        def regenclicked(self, widget):
2054            t = ObjetoBase.mac.genmac()[1].split(":")
2055            for i in t:
2056                tmpentry = self.link.builder.get_object("chg_MAC-entry" + str(t.index(i)))
2057                tmpentry.set_text(i)
2058                tmpentry.show()
2059
2060    class about(Gtk.AboutDialog):
2061        def __init__(self):
2062            self.win = builder.get_object("AboutWindow")
2063            self.win.connect("delete-event", self.destroy)
2064            self.win.connect("response", self.destroy)
2065            self.win.add_credit_section("Tutores", ["Julio Sánchez"])
2066            #self.win.add_credit_section("Contribuidores", [""])
2067            self = self.win
2068        def show(self, *args):
2069            print("Showing")
2070            self.win.show()
2071        def destroy(self, *args):
2072            self.win.hide()
2073            return True
2074
2075
2076    #Esta clase te permitirá deshacer acciones, algún día de un futuro lejano.
2077    class Undo():
2078        def __init__(self):
2079            self.lastactions = []
2080
2081    #Esta la pongo fuera porque lo mismo la necesito en otra clase
2082
2083    def exiting(self, *ahfjah):
2084        global log
2085        savelog()
2086        lprint("End time: " + time.strftime("%H:%M:%S"))
2087        print ("Window closed, exiting program")
2088        Gtk.main_quit()
2089
2090    def restart(*args):
2091        global log
2092        savelog()
2093        lprint("End time: " + time.strftime("%H:%M%S"))
2094        lprint("Restarting program")
2095        print("\033[92m################################\033[00m")
2096        os.chdir(startcwd)
2097        os.execl(sys.executable, sys.executable, *sys.argv)
2098
2099    def returnTrue(*lala):
2100        return True
2101
2102    def nothing(self, *args):
2103        #Funcion Hugo
2104        pass
2105
2106    def leppard():
2107        lprint("Gunter glieben glauchen globen")
2108
2109    writeonlog("Esto ha llegado al final del codigo al parecer sin errores")
2110    writeonlog("O tal vez no")
2111    MainClase()
2112
2113    #end()
2114
2115    lprint("Actual time: " + time.strftime("%H:%M:%S"))
2116    lprint("Complete load time: " + str(datetime.now() - startTime))
2117    push_elemento("Parece que esta cosa ha arrancado en tan solo " + str(datetime.now() - startTime))
2118    Gtk.main()
2119
2120    print("\033[92m################################\033[00m")
```

B.2. modules/Save.py