

# Proxy caché Squid en Raspberry Pi

**Francisco José de Torres Sánchez-Simón**

## **Índice:**

### **1. Análisis y diseño del sistema:**

- 1. El objetivo**
- 2. El proxy caché**
- 3. Mi caso particular**

### **2. Análisis de la interfaz:**

- 1. El proxy caché en una Raspberry Pi**

### **3. Manual del Administrador**

- 1. El sistema operativo en la Raspberry Pi**
- 2. La instalación del sistema operativo en la Raspberry Pi**
- 3. La instalación del proxy Squid**
- 4. La configuración del sistema para que el proxy sea transparente al usuario**

### **4. Manual del usuario**

# Análisis del Sistema I

## El Objetivo:

La limitación de ancho de banda que existe en los institutos de secundaria es algo que he observado tanto en Extremadura, donde inicié el ciclo y cursé el primer año completo, como en Andalucía en mi segundo año. Existe un ancho de banda que es limitado y disponible para buena parte de los alumnos, si no todos para realizar su navegación web y descarga de (en teoría) de contenido necesario para el seguimiento de las clases. Estamos hablando por lo tanto de la necesidad de descargar software específico para instalar en los ordenadores para el seguimiento de las asignaturas. Por otro lado, los sistemas operativos que se utilicen necesitan descargas periódicas de actualizaciones de los mismos. Cada ordenador realizando una misma descarga satura realmente la conexión a internet en muchísimos casos. Teniendo en cuenta que este es un problema real que sucede habitualmente, se antoja absurdo que tengan que ser descargados desde internet los mismos datos tantas veces.

Por otro lado, el contenido “de ocio” al que accedemos a veces los alumnos que, aunque no deberíamos en quizás la mayoría de los casos, sí es real que se produce, también supone una disminución del ancho de banda disponible para el conjunto de usuarios de la red que puede acceder a internet. Por contenido “de ocio” podemos entender toda aquella navegación que no está relacionada con el seguimiento de las clases, como podrían ser las redes sociales y portales de contenido multimedia; entre los que destaca, como no, *youtube*.

Una solución a este último problema de colapso del ancho de banda por acceder a contenido no necesario para seguir las clases, puede resolverse, al menos a priori, de dos maneras: controlando y denegando el acceso a contenido específico mediante algún software, o permitiendo que se almacene este contenido, susceptible de ser poco cambiante en caso de muchos elementos multimedia, en la red local para que no sea necesario que vuelva a ser descargado una y otra vez.

Para ambos cometidos el tipo de software adecuado sería:

## El proxy Cache:

Como bien se sabe, un proxy cache es un servicio, generalmente alojado en una máquina específica, que sirve de conexión intermediaria entre los clientes que acceden a otros ciertos servidores (o toda la internet globalmente), y estos destinatarios finales.

Si el proxy tiene la capacidad de almacenar contenido en memoria (llamada “cache”) que pueda luego servir a sus clientes para evitar nuevas conexiones, se le denomina, precisamente, “proxy cache”. Si además en las conexiones en las que hace de intermediario, no se identifica a sí mismo como quien las solicita, sino que tan sólo transfiere las peticiones recibidas desde sus clientes, ocurriendo así que el servidor de destino identifica a estos últimos como quienes realizan estas últimas; se les denomina “proxy transparente”. Es decir, el proxy hace las peticiones de conexión (http, ftp y/u otros), pero no se identifica lo más mínimo.

Existen otra clase de proxies, como el inverso, pero nos interesa tan sólo ahora definir qué clase de proxy es el que queremos instalar nosotros para los propósitos antes descritos.

En el caso de que la opción sea bloquear páginas web específicas para evitar, por ejemplo, la

descarga de contenido en streaming, puede no siempre ser conveniente, puesto que, al menos este segundo año, en algunas ocasiones hemos necesitado acceder a vídeos de **youtube** con contenido relativo a lo que estábamos estudiando. Así pues, bloquear *youtube*, en principio, no es la solución ideal para evitar tanto tráfico generado al visitar ese portal. Se ha dado el caso, por ejemplo, de estar viendo varios de nosotros un mismo vídeo relativo a seguridad informática, en inglés, como parte de un ejercicio en las clases de inglés. En este caso, así como en otros en los que la necesidad consistía en descargar todos un mismo archivo de instalación de un software específico, resulta absurdo que se realicen múltiples descargas de un mismo contenido si eso pudiera ser evitable.

Cuando se trata de descarga de archivos, el problema de colapso de la conexión a internet se arreglaba si el profesor ya lo había descargado, o si era el primero en hacerlo, o si era un alumno quien lo había descargado primero y a partir de ahí se compartía en la red local a través del protocolo NETBIOS. En Windows, por defecto, se permiten hasta un máximo de 10 conexiones simultáneas a un recurso compartido, y suele haber en las aulas, desde luego, más de 10 alumnos. Eso ha supuesto un problema real que ralentizaba el transcurso de la explicación, puesto que ha habido veces en las que los archivos a descargar eran grandes.

Este método de que alguien descargue una copia la y la comparta en la red local no es la solución para, por ejemplo, los paquetes de un sistema operativo que se actualiza. Todos los CD/DVD de instalación de cualquier sistema operativo tiene sus paquetes desactualizados porque, desde que fueron creados los originales, han salido diversos parches de actualización que, una vez más, cada cliente tendrá que descargar desde internet. Si todos los alumnos están probando a instalar, pongamos por caso, la distribución linux especializada en seguridad “Kali Linux”, se encontrará cada uno con que la actualización son algo más de 1 GB.

Se hace pues, muy interesante, si no imprescindible, poder almacenar en cache, en un proxy local, todo el contenido que se prevea que va a ser descargado desde internet directamente, múltiples veces.

## Mi caso particular:

He sido usuario habitual de linux en los dos años de curso del módulo superior A.S.I.R. en el aula. Es decir, mientras mis compañeros usaban Windows para su quehacer diario, yo hacía lo mismo desde linux. Eso implicó tener que usar algunos programas nativos de Windows a través de *wine*; como el propio *packet tracer* de CISCO. Así como su curso instalable en formato html con contenido flash. En casa usaba, generalmente, la misma distribución de Linux, que solía ser un *Kubuntu*.

Soy consciente de que actualizar un sistema linux puede ser una carga pesada para la conexión a internet, y había días, además, que resultaba casi imposible debido al tráfico intenso al que a veces estaba sometida. Mi necesidad fue, por tanto, poder portar los paquetes descargados en casa por una actualización del sistema o por la instalación de algún tipo de contenido. Para el caso de los ordenadores del aula tenía ya preparados una serie de scripts “completers”, como me gustaba denominarlos, para personalizar al máximo mi distribución linux con las aplicaciones que suelo utilizar habitualmente.

Opciones para resolver el problema:

1. Copiar el contenido de los paquetes descargados en `/var/cache/apt/archives/*.deb` a un pendrive o algún disco duro externo y pasarlos a mi sistema de clase. Ejecutar un `sudo apt-get update` para que mi PC de clase tenga en cuenta esos nuevos paquetes en su cache interna.
2. Utilizar el programa APTONCD, que crea una imagen iso con todos los paquetes anteriores

y que tendría que tener instalado en ambos sistemas; lo que por otra parte no supone un problema. Téngase en cuenta que APTONCD requiere, para que funcione correctamente y podamos montar una iso creada por él mismo en otra máquina, el paquete *hal*. Por lo que la instalación correcta de APTONCD sería así:

```
sudo apt-get install hal aptoncd
```

3. Cachear el contenido de lo descargado en casa en un proxy caché “portátil” que pueda llevar a clase. Es ahí donde encuentra protagonismo nuestro proyecto.

Por otra parte, también me ha sido necesario descargar cientos de bytes durante este curso en forma de service packs para Windows y programas de diverso tipo. Eso sí que es factible que pueda descargarse en casa y ser traído en un *pendrive*.

Las actualizaciones de Windows, son quizás algo complejas de extraer para llevarlas “en frío” a otro sistema, y, en cualquier caso, es un método más engorroso que la solución adoptada.

# Análisis de la Interfaz

## Proxy Cache en una Raspberry Pi

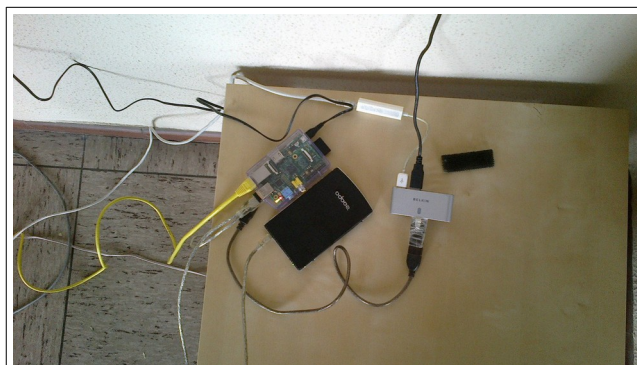
La Raspberry Pi es un nano ordenador tan sumamente pequeño que cabe en la palma de una mano. Cuenta, a grandes rasgos, con un microprocesador ARM de 700 Mhz (el equivalente en potencia a los primeros Samsung Galaxy, que tienen micros de esa misma arquitectura); una RAM de 512 MB, una GPU que permite salida de vídeo HD a través de un conector HDMI (tiene salida HDMI), y salida por vídeo compuesto y canal de audio. Cuenta además con puertos GPIO que permite la entrada/salida de señales analógicas, lo que permite usar la Raspberry Pi de manera similar a la placa *Arduino*, ya que a través del puerto GPIO se puede comunicar con el exterior.

Cuenta también con un puerto *ethernet* que a nosotros nos va a ser especialmente útil, y dos puertos USB 2.0.

Estamos hablando, dicho sea de paso, de la versión B del producto, puesto que la primera versión, aún a la venta y más barata, cuenta con tan sólo 256 MB de RAM y no tiene puerto *ethernet*. Esta carencia puede ser suplida con un adaptador USB-ethernet, que además necesitaremos para nuestro propósito; pero creo que ese gasto extra por la necesidad surgida de un adaptador ethernet para USB es más o menos la misma diferencia de precio que hay entre los modelos A y B, y, por otra parte, liberamos un puerto USB.

Ambos modelos cuentan además con una ranura para tarjetas SD. Este es un elemento crucial en este dispositivo, puesto que ha sido diseñado para buscar las instrucciones de arranque, precisamente, en la tarjeta de memoria SD. Se puede instalar el sistema operativo en una unidad externa en un USB, pero es necesaria la presencia de la tarjeta de memoria porque el aparato está diseñado para buscar, entre otras cosas la ruta del sistema de arranque. Esto lo veremos en detalle en el apartado *Manual del Administrador*.

No lleva ventilación propia: la ventilación del sistema es, pues, pasiva. Su alimentación es a través de una fuente de 5V y una intensidad de corriente entre 1200 y 2000 mA. Esto significa que, aunque se fabrican fuentes de alimentación propias para este dispositivo (yo cuento con una de 2000 mA), sirven las de algunos móviles y, lo que es más interesante, también una salida USB mínimo 2.0 de nuestro ordenador de sobremesa o portátil.



a



b



c



d

d

Distintas configuraciones para la Raspberry Pi. En a.), b.) y d.) Se está utilizando un disco duro externo que alberga el sistema operativo, en vez de la tarjeta SD necesaria para el arranque. En c.) configuración definitiva del proxy cache transparente al usuario, donde el disco duro externo se ha sustituido por un *pendrive* y ya no es necesaria la alimentación adicional.

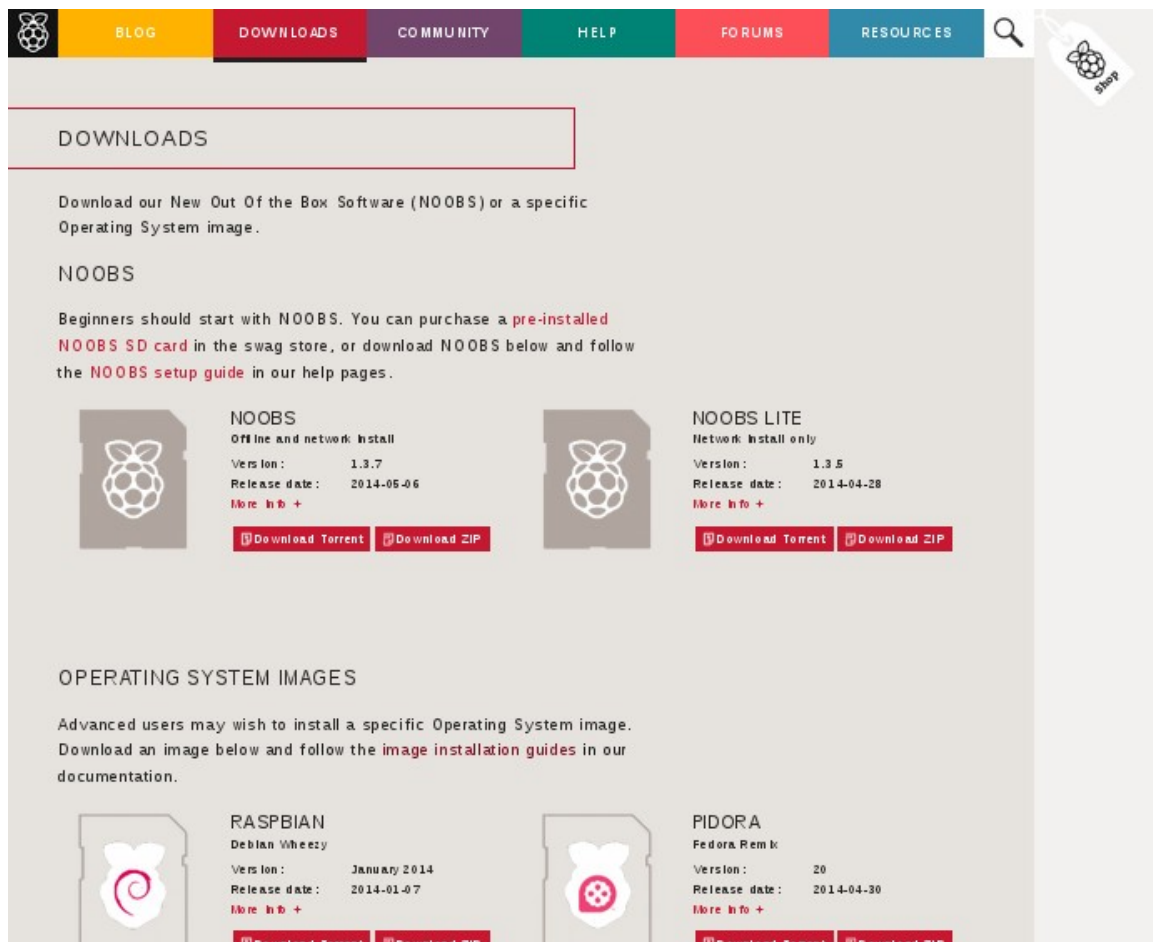
El hecho de que su ventilación sea pasiva y sus necesidades energéticas tan bajas, la hace ideal para tenerla en funcionamiento permanentemente al igual que un *router* casero, por ejemplo.

El sistema operativo que la hace funcionar es cualquier versión de linux compilado para la arquitectura ARM, y, concretamente, variantes de Debian especialmente optimizadas para Raspberry Pi.

# Manual del Administrador


## El sistema operativo en la Raspberry Pi:

<http://www.raspberrypi.org/downloads/>



La idea inicial fue que la Raspberry Pi fuera una Smart TV a través del proyecto de software libre [XBMC](#). Cuyo mayor exponente es, por su facilidad de uso, por tratarse de una versión de Debian optimizada para la Pi y con el XBMC ya instalado, el proyecto [Raspbmc](#):





[About](#)
[Wiki](#)
[Forum](#)
[Download](#)
[Kits](#)
[Contact](#)
[FAQ](#)
[Donate](#)

---

## Raspbmc's June update

Hi

Raspbmc's June update is small, but sweet. Here's what's new:

- XBMC 13.1 final with a ton of fixes for some of the bugs introduced in Gotham:
  - Fix resume point of certain PVR items
  - Fix issues with UPNP
  - Fix crash after AirPlay finishes
  - Fixes for FLAC tracks with embedded images
  - Fixes for M4A and MP4 playback
  - Fix a minor memory leak
  - Use a unique client broadcast ID in the PVR API
- Some fixes for improving support using external soundcards (HifiBerry and iqaudio), see this [thread here](#)
- Reverted to ext4 as the default filesystem for Raspbmc. ext4 provides better data redundancy and is more compatible with Linux systems for those wishing to backup or transfer data.
- Fix issue where XBMC will keep telling user a new version is available.
- Fixes for some issues with 3D content

### About

Welcome to the home of Raspbmc, the simplistic yet powerful Raspberry Pi media center distribution, created by Sam Nazarko!

[Donate](#)

---

### News


[Raspbmc's June update](#)

[Raspbmc's May update](#)

[XBMC 13 'Gotham' is here!](#)

---

### Social



El problema encontrado cuando se probó esta distribución era que realmente mantener un entorno gráfico en ejecución era poco práctico debido al consumo de recursos de éste, en conjunción con el que iba a ser nuestro servicio estrella: el proxy cache *squid*.

Lo primero que se intentó implementar fue que la Raspberry actuara como punto de acceso a través de una tarjeta wi-fi por USB adquirida en el momento de la compra (por ser, supuestamente, especialmente compatible con nuestro dispositivo), cuyo modelo es una [Edimax EW-7811Un](#)

Tras muchos intentos frustrados con Raspbmc como sistema operativo base, descubro en algún foro que más de un usuario comenta que con esta variante de Debian para Raspberry Pi no funcionan las numerosas instrucciones de blogs y tutoriales que hay en internet. Pero tampoco se conoce, o al menos no he sido capaz de encontrar, su causa.

Solución: cambiamos a *Raspbian*, sobre todo porque al haber desechado la idea de tener una *Smart TV* y centrarnos en el objetivo inicial, un sistema que no requiera de entorno gráfico, tendría un rendimiento mucho mayor.

En *Raspbian*, afortunadamente, sí que funcionan nuestro primer objetivo: que la Raspberry Pi sea un punto de acceso al que poder conectarnos. El tutorial elegido es éste:

<http://www.daveconroy.com/turn-your-raspberry-pi-into-a-wifi-hotspot-with-edimax-nano-usb-ew-7811un-rtl8188cus-chipset/>

# Turn Your Raspberry Pi Into a WiFi Hotspot with Edimax Nano USB EW-7811Un (RTL8188CUS chipset)

Posted by dconroy on Jul 10, 2013 in How To's, Raspberry Pi | 125 comments | 83,848 views

I'm writing this blog to help anyone with an Edimax Nano USB WiFi adapter (EW-7811Un) configure a Wireless Access Point. The main reason I wrote this post is because the Edimax was the first wireless adapter I ever recommended (turning a Raspberry Pi into a fully functional web server). I chose this adapter because it works out of the box, is cheap, and has a really low profile. But when I started the process of trying to turn my rPi into a WiFi hotspot, it seemed that every tutorial out there claimed that the Edimax card I had recommended "doesn't support Access Point".

**Luckily for you and me, this is not the case.**



El resultado, aún así, no me tiene del todo contento, puesto que la mayoría de las veces nos es imposible conectarnos por vía ssh al aparato, teniendo que hacer trucos como desconectar el adaptador wifi, el cable *ethernet*, esperar unos 10 segundos y volver a conectar tan sólo el cable *ethernet*.

Se trata, por si no se ha apreciado todavía, de evitar a toda costa necesitar un monitor HDMI y teclado y ratón USB con el que acceder al sistema, pretendiendo una conexión a través de ssh en todo momento.

Tanto *Raspbmc* como *Raspbian* cuentan con un servidor ssh instalado por defecto, que en ambos casos es el *openssh-server*; aunque no es el único disponible, como veremos ahora.

El tutorial anterior propone conectar los clientes del punto de acceso, que conectarían a la tarjeta inalámbrica, con la red local que nos da servicio, a través de una conexión puente, que se consigue, en los sistemas linux, instalando el paquete *bridge-utils*.

En otros casos la solución sería instalar un servidor DHCP para crear una subred distinta a la que estén conectados los clientes unidos a la tarjeta inalámbrica, que tendría esta también una IP en esa misma subred. La tarjeta *ethernet* que nos conecta a la LAN en la que está el *router* que nos da salida a internet, tendría una IP en el rango de la subred creada por el *router*.

A mí no me gustaba nada la idea de tener a los clientes conectados inalámbricamente, a una subred distinta de la inicial, pese a que una implementación de reglas con *iptables*, que es parte necesaria en esta configuración, nos comunicara ambas redes como si de otro *router* se tratara.

## La instalación de un sistema operativo en Raspberry Pi

Como puede observarse en la zona de descarga de la dirección <http://www.raspberrypi.org/>, el contenido de un sistema operativo *linux*, sea de la distribución que sea, se obtiene a partir de una imagen (archivo img) listo para ser volcado en una memoria SD de al menos 4Gb.

La distribución *Raspbmc* cuenta con un instalador tanto para *linux* (que en este caso es un script), como para *Windows*, que en este último caso se trata de un programa de uso muy intuitivo. Para esta distribución nos libramos de descargar una imagen que tengamos que saber volcar por nuestra cuenta sobre la SD; tarea que, por otra parte, solo podría antojarse compleja para usuarios no entendidos en la materia. Nosotros estamos en disposición de administrar un sistema *linux* con un *proxy caché* en ejecución, por lo que volcar, desde *linux* por ejemplo, la imagen en nuestra tarjeta de memoria, no debería ser problema.

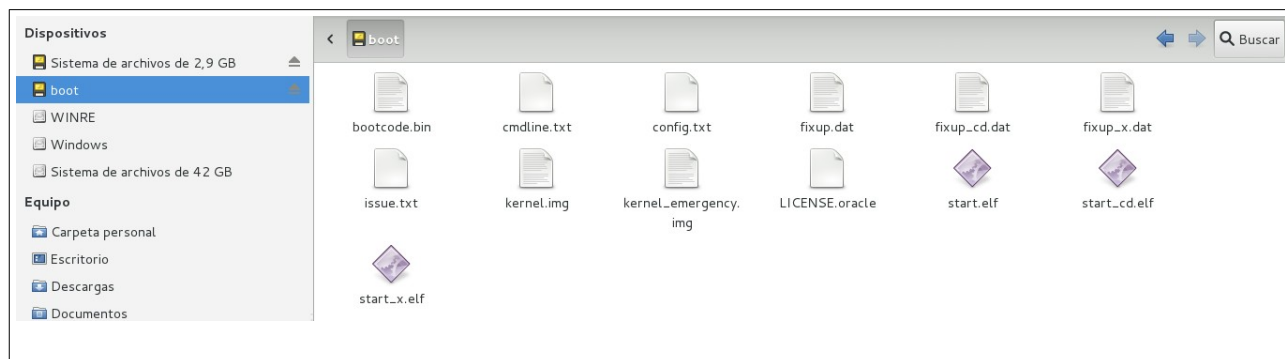
De hecho, y para el caso de *Raspbian*, que fue la siguiente distribución en fase de pruebas, es, si el archivo de imagen tiene como nombre *2013-10-13-wheezy-minibian.tar.gz*

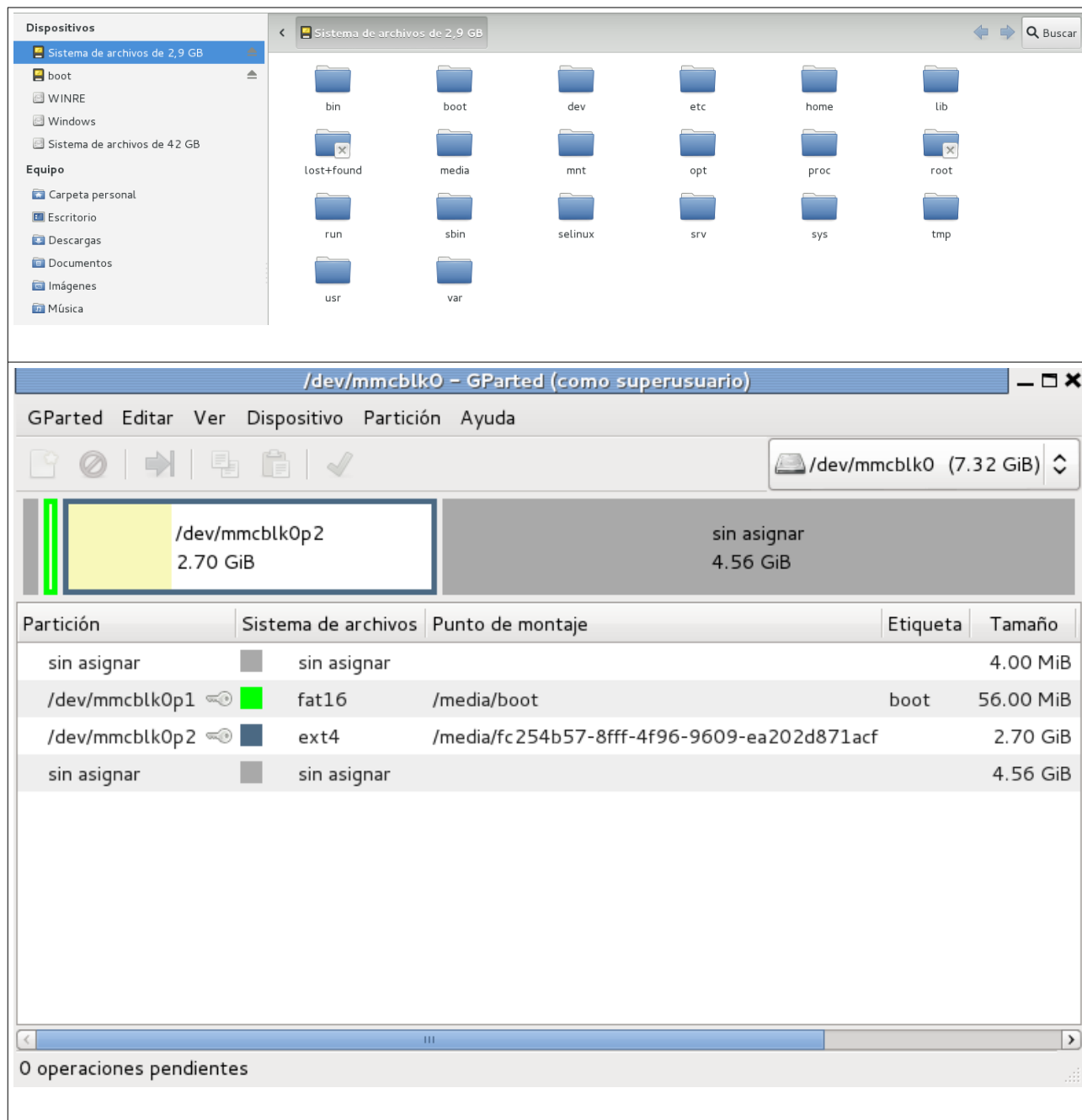
**tar xvzf 2013-10-13-wheezy-minibian.tar.gz** #pues tenemos que desempaquetar la imagen, que se distribuye comprimida.

**sudo dd if=2013-10-13-wheezy-minibian.img of=/dev/mmbk0 bs=1M**

Que significa hacer un volcado con el comando *dd* teniendo como archivo de origen la imagen *2013-10-13-wheezy-minibian.img*, y como destino la tarjeta en caso de que haya sido reconocida como *mmbk0* si se tiene un lector de tarjetas integrado en el ordenador. En caso de que haya sido leída a través de un adaptador USB, será reconocida como */dev/sdx*, siendo x, a,b,c... según proceda en cada caso. Téngase en cuenta que no volcamos a una partición del dispositivo SD, sino a ella, por lo que no se hace a */dev/sda1*, o */dev/mmbka0p1*, etc.

Tanto el método de instalación a través del script fácil de *Raspbmc*, como el método de volcado de imagen de *Raspbian*, nos dejan al menos una partición *fat32* en el primer caso, y una segunda partición (en el segundo caso) tipo *ext4* que sólo es vista desde *Linux* o *Mac OS X*. La tarjeta siempre contará con esas dos particiones. En una, la *fat32*, estarán los archivos que busca la *Raspberry Pi* cuando arranca, y en la segunda el sistema *linux* instalado.





Partición fat32 y Ext4 generadas en la tarjeta de memoria. Obsérvese que, pese a que en nuestro caso, la tarjeta tiene una capacidad de 8GB, el sistema linux está sobre una partición de 2,9 GB, quedando el resto sin asignar, como puede verse a través de Gparted.

El archivo responsable del arranque es cmdline.txt. Es editable en texto plano y su contenido inicial es:

```
dwc_otg.lpm_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200 console=tty1
root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
```

En él, un aspecto que nos interesa mucho es lo señalado en **negrita**: dónde busca el sistema de archivos *linux*. En este caso irá a la segunda partición de la SD, reconocida como */dev/mmcblk0p2*.

Para instalar el sistema en, por ejemplo, un *pendrive* o un disco duro externo, deberemos insertarlo en la *Raspberry Pi* cuando esté funcionando y ver de qué manera lo ha reconocido.

Generalmente será `/dev/sda1`, en el caso de que tenga una partición. En las opciones anteriores, si cambiamos `root=/dev/mmcblk0p2` por `root=/dev/sda1`, habremos conseguido que el arranque se efectúe desde el dispositivo elegido, que es donde habremos instalado nuestro sistema.

¿Por qué nos puede interesar una configuración así y perder, por tanto uno de los dos puertos USB libres?

Porque la experiencia de muchos usuarios con *Raspberry Pi* antes que nosotros han advertido de que, en configuraciones donde se hace un uso exhaustivo de la SD a base de continuas escrituras sobre ella, éstas acaban fallando y lo perdemos todo. Y podemos prever que será nuestro caso, puesto que queremos instalar un *proxy* y eso implica una escritura constante de *cacheo* de contenido de navegación. Necesitamos pues, hacer una configuración de sistema operativo en dispositivo externo.

Otra opción, si no queremos trasladar el sistema operativo entero a otra unidad, es montar los directorios de trabajo de *squid*, y de cualquier otro programa que haga un uso intensivo del medio de almacenamiento, en un dispositivo USB. Nos ha interesado más que esté todo alojado directamente en una memoria USB.

## El proceso completo de instalación

En nuestro periplo por las distintas versiones de Debian para *Raspberi Pi*, *Raspbian* no fue, desgraciadamente, nuestra parada final. Adolecía de un peso excesivo cuando conseguimos configurarlo como punto de acceso a través de una conexión puente entre la tarjeta de red *ethernet* y la inalámbrica, habiendo lanzado también *squid3* y un servidor *trasmision* para descargas de archivos *torrent*. Del servidor *squid3* se puede decir que, en esos días, había sido instalado de la manera fácil: a través del gestor inteligente de paquetes *apt* y desde entonces funcionaba con su configuración por defecto, es decir, prácticamente ninguna. Simplemente, estaba en ejecución.

Al descubrir que había una variante de *Raspbian* “descafeinada” porque se le había quitado muchos paquetes considerados no imprescindibles para el arranque básico del sistema, nos decantamos por esta última; y es la que actualmente soporta el proyecto integrado. De *Raspbian*, decir, por último, antes de olvidarnos de ella, que tiene la opción de cargar el entorno gráfico desde un principio, o lanzarlo a través de la orden *startx*. Para nuestros propósitos, elegimos la última opción; pero el entorno gráfico *LXDE* ya está instalado.

## La instalación de Moebius

Optamos finalmente por esta distro “light” de Debian: <http://moebiuslinux.sourceforge.net/>

La descarga de la imagen comprimida nos deja un archivo llamado, en nuestro caso: *moebius-1.1.1.tar.gz*, que se descomprime, como antes, con:

```
tar -xvzf moebius-1.1.1.tar.gz
```

Y pasamos la imagen a la SD a través de:

```
sudo dd if=moebius-1.1.1.img of=/dev/mmcblk0 bs=1M #bs=1M significa que se transfieren los bits en bloques de un megabyte
```

A continuación se introduce la tarjeta SD a la *Raspberry Pi* que deberá estar además conectada a internet a través de su puerto *ethernet* y contar con alimentación para funcionar.

Si disponemos de una pantalla con conexión HDMI y teclado y ratón por USB, podremos ver el proceso de instalación completo, puesto que la *Raspberry Pi* se reiniciará varias veces. Lo que está ocurriendo ahí, es que se está creando la partición Ext4 y se está instalando el sistema en él.

El resultado final es que podremos conectarnos remotamente con ella a través de *ssh*, puesto que viene por defecto un servidor *ssh* instalado. Tendremos que hacer intentos de conexión sucesivos hasta saber si el servidor ya está a la escucha. Esto ocurre cuando el proceso de instalación ha terminado. En esta distro lo que pasa es que el proceso acaba con una pantalla de configuración en modo texto (*raspi-config*) que realmente no se llama así pero es igual, y si queremos realmente volver a ejecutar esa herramienta, la tendremos que descargar de los repositorios. Si, cuando presumimos que la instalación ya ha terminado, no obtenemos conexión por *ssh*, podemos desenchufar el equipo y volverlo a enchufar para que arranque de nuevo. Esto es una manera sucia de apagar el sistema, deberá hacerse siempre mediante algún comando, pero, si no tenemos una pantalla con puerto HDMI, además de teclado y ratón USB; no tendremos ninguna otra alternativa y un reinicio fue, en nuestro caso, necesario para poder entrar por *ssh* y completar su instalación.

Otro asunto a tener en cuenta es que en la instalación la ip asignada a la tarjeta *ethernet* es por DHCP y a priori no sabemos cuál es, al no ser que tengamos un monitor, teclado y ratón. En caso de que no, podemos, desde *linux* (o *Windows*), usar *nmap* como *root* para detectar la presencia y la IP de la *Raspberry Pi*:

**nmap -sP 192.168.1.0/24** #En caso de que nuestra subred sea la indicada. Miramos donde aparece el nombre *Raspberry Pi Foundation*, y, dos líneas anteriores es la IP de nuestra *Raspberry*.

Ahora podemos hacer:

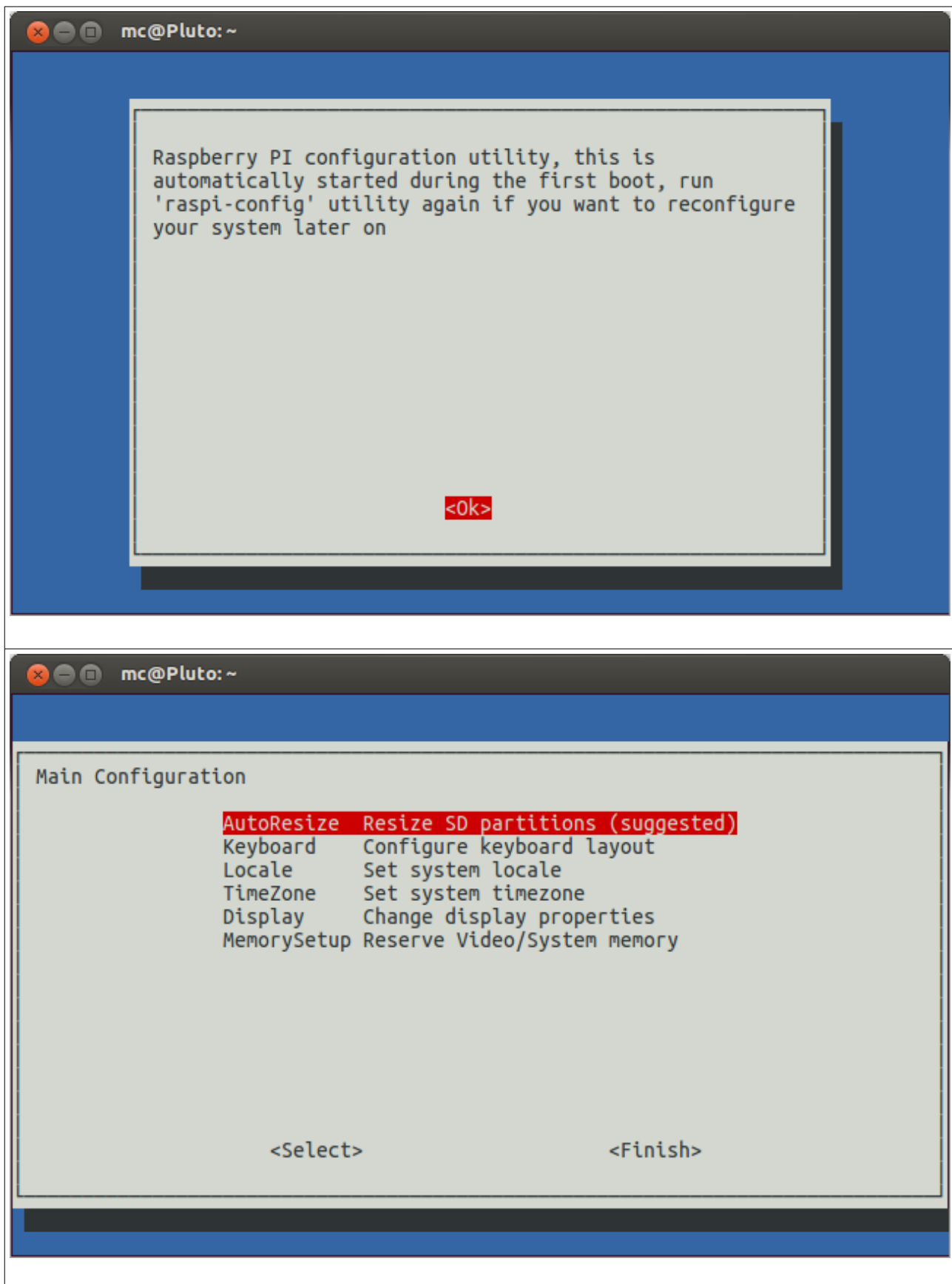
**ssh root@ip raspberrypi** #siendo la contraseña por defecto, *raspi*. En la mayoría de las distribuciones, el usuario que viene por defecto es *pi* y contraseña *raspberrypi*.

Para tener esa otra configuración más habitual, podremos crear el usuario *pi* con el comando interactivo **adduser pi**, que generará un directorio de trabajo en */home/pi*. Pero antes de eso deberíamos editar el archivo */etc/default/useradd* y cambiar **SHELL=/bin/sh** por **SHELL=/bin/bash**, para que sea esa la shell por defecto que se asigne a cada nuevo usuario creado.

Lo que viene a continuación ha sido parcialmente obtenido de las siguientes fuentes:

<http://www.ubuntumax.com/2013/01/el-home-server-perfecto-con-una.html>  
<http://www.raspberrypi.org/forums/viewtopic.php?f=29&t=44177>

En el primer inicio, tanto desde pantalla como desde *ssh*, nos encontraremos con esto:





Después de las fotos anteriores podemos entender qué significa la primera opción de modificar el tamaño de la partición: La SD no está ocupada del todo al no ser que hayamos usado una de 4GB y nos propone extenderla para utilizar todo el espacio. Las otras 3 opciones son la configuración del teclado, el idioma local a usar y la hora. La cuarta opción es para el caso de usar una pantalla HDMI o de vídeo compuesto (uso de pantalla en todo caso) y es algo que, para el uso concreto que le vamos a dar, no nos interesa. El redimensionado del sistema de archivos para que ocupe toda la SD tampoco nos va a hacer falta porque vamos a trasladar todo el sistema a un *pendrive* o disco duro externo.

La última sí: es la cantidad de memoria asignada a la GPU. Lo mínimo es darle 16 MB y esa fue nuestra elección para disponer del máximo para los servicios que haríamos correr.

Una vez creado el usuario *pi*, le asignamos una contraseña a través de:

**passwd pi**

Probamos que está en *sudoers* a través de *sudo pi ifconfig* (por ejemplo)

Anulamos la cuenta root por seguridad, como se hace en *ubuntu*, con el comando:

**sudo su**

**passwd -l root**

A continuación resulta conveniente pasar ya el sistema a, pongamos por caso, un *pendrive* con suficiente capacidad. El utilizado en fase de pruebas por mí para el proyecto integrado es de 8 Gb, y en *squid* se ha configurado un uso de disco para cache de 5 Gb.

Para ello, apagamos el sistema con un **sudo halt**, extraemos la SD y la insertamos en el ordenador para poder leer en ella. Los sistemas *Windows* sólo detectarán la partición *fat32* en la que está el archivo *cmdline.txt* que es ahora el único que nos interesa.

En éste cambiamos *root=/dev/mmcblk0p2* por *root=/dev/sda1* si es de esa manera como se detectó por parte del sistema la unidad de memoria por USB insertada (según se ha explicado ya).

Además, introduciremos, al final del archivo, *rootdelay=5* para que el sistema deje 5 segundos de espera antes de acceder al dispositivo USB, puesto que a veces puede darse una cierta latencia por parte del hardware.

El siguiente paso es transferir todo el sistema de archivos generado en la SD (*Ext4*) al *pendrive*. Una manera, más lenta pero al alcance de usuarios *Windows*, es, desde la propia *Raspberry Pi*.

Necesitamos formatear nuestro *pendrive* en *Ext4*, para lo que necesitamos un sistema *linux* con *Gparted* (por ejemplo) que nos haga la tarea fácil. Una vez preparado el *pendrive* se inserta y miramos cómo ha sido detectado en el sistema con un:

**sudo fdisk -l** #Y deberíamos ver alguna mención a */dev/sda*

hacemos lo siguiente:

**sudo mount /dev/sda1 /mnt/** #Lo montamos en */mnt* que suele estar en desuso

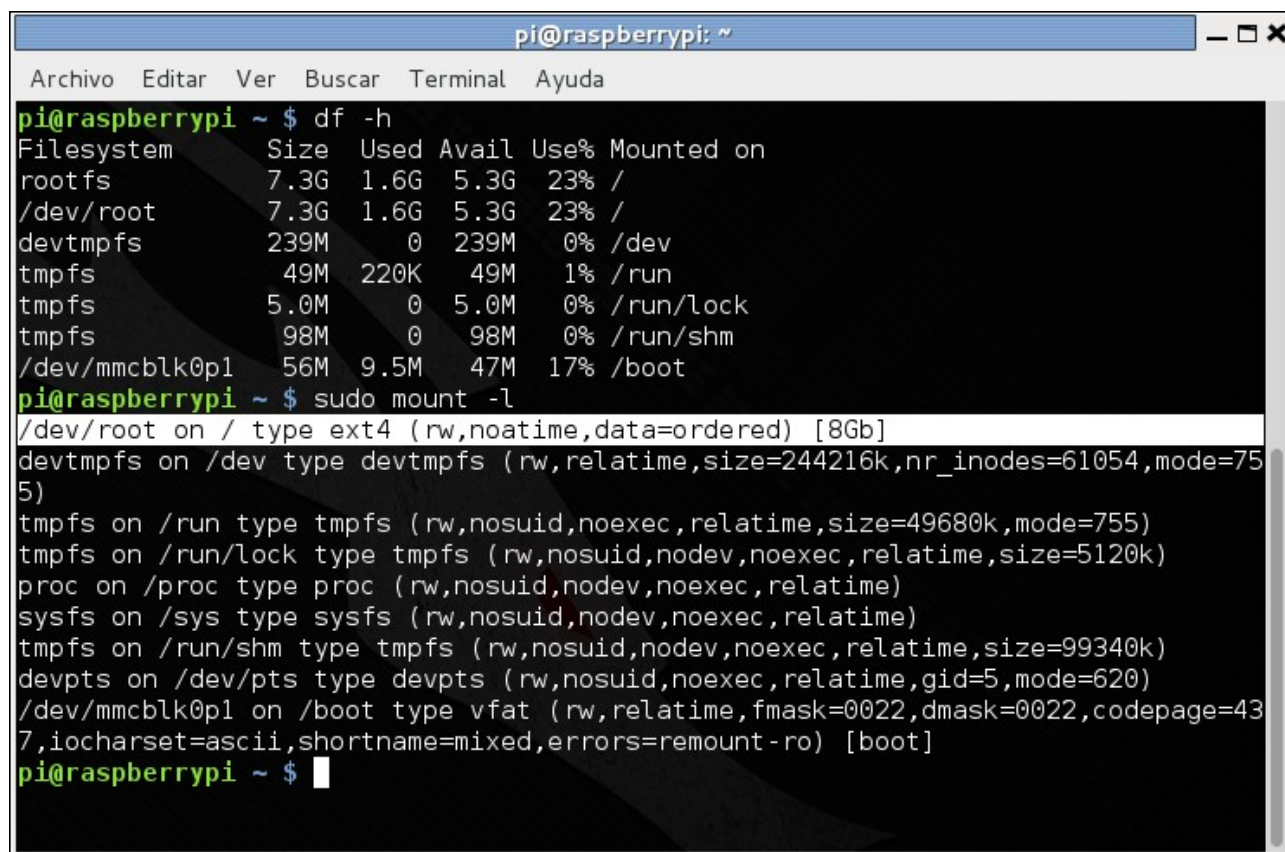
**sudo rsync -xav / /media/** #Transferimos todo el contenido preservando toda clase de atributos de los archivos. Especialmente importante preservar los permisos, dueños, etc.



editamos /mnt/etc/fstab y lo cambiamos para que quede así:

```
proc          /proc          proc defaults      0    0
/dev/mmcblk0p1 /boot          vfat defaults      0    2
#/dev/mmcblk0p2 /              ext4 defaults,noatime 0    1
/dev/sda2 /          ext4 defaults,noatime 0    1
# a swapfile is not a swap partition, so no using swapon|off from here on, use dphys-swapfile
swap[on|off] for that
```

Podemos reiniciar con `sudo reboot` y comprobar que estamos hemos arrancado desde el *pendrive* si preguntamos a `mount` dónde está cargado el sistema de archivos raíz:



```
pi@raspberrypi: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
pi@raspberrypi ~ $ df -h
Filesystem      Size  Used Avail Use% Mounted on
rootfs          7.3G  1.6G  5.3G  23% /
/dev/root        7.3G  1.6G  5.3G  23% /
devtmpfs        239M   0    239M   0% /dev
tmpfs           49M    220K  49M    1% /run
tmpfs           5.0M   0    5.0M   0% /run/lock
tmpfs           98M    0    98M    0% /run/shm
/dev/mmcblk0p1  56M   9.5M  47M   17% /boot
pi@raspberrypi ~ $ sudo mount -l
/dev/root on / type ext4 (rw,noatime,data=ordered) [8Gb]
devtmpfs on /dev type devtmpfs (rw,relatime,size=244216k,nr_inodes=61054,mode=755)
tmpfs on /run type tmpfs (rw,nosuid,nodev,noexec,relatime,size=49680k,mode=755)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /run/shm type tmpfs (rw,nosuid,nodev,noexec,relatime,size=99340k)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620)
/dev/mmcblk0p1 on /boot type vfat (rw,relatime,fmask=0022,dmask=0022,codepage=437,iocharset=ascii,shortname=mixed,errors=remount-ro) [boot]
pi@raspberrypi ~ $
```

Ahora procede hacer un:

**apt-get update && apt-get upgrade -y && apt-get dist-upgrade -y**

Que nos dejará el sistema actualizado.

## La instalación del proxy Squid

La instalación procederá tal y como indica el siguiente enlace (funciona todo tal y como está aunque con alguna modificación):

<http://aacable.wordpress.com/2014/04/21/howto-cache-youtube-with-squid-lusca-and-bypass-cached-videos-from-mikrotik-queue/>

La idea es la siguiente: vamos a usar un “fork” o variante del *squid* (versión estable 2.7) y para ello vamos a compilarlo desde el código fuente. Es una suerte que esté a disposición el código fuente teniendo en cuenta que nuestra arquitectura es ARM y no i386. En la *Raspberry Pi*, la compilación a funcionado perfectamente. La razón por la que se usa esta variante específica es porque es un proyecto surgido específicamente para poder cachear contenido de *youtube*, entre otros de contenido de url cambiante.

Lo primero que hacemos es:

```
sudo su #Imprescindible convertirse en root
apt-get install gcc -y &&
apt-get install build-essential -y &&
apt-get install libstdc++6 -y &&
apt-get install unzip -y &&
apt-get install bzip2 -y &&
apt-get install sharutils -y &&
apt-get install ccze -y &&
apt-get install libzip-dev -y &&
apt-get install automake1.9 -y &&
apt-get install acpid -y &&
apt-get install libfile-readbackwards-perl -y &&
apt-get install dnsmasq -y &&
mkdir /lusca &&
cd /lusca &&
wget -c
http://wifismartzone.com/files/linux\_related/lusca/LUSCA\_HEAD-r14942.tar.gz &&
tar -xvzf LUSCA_HEAD-r14942.tar.gz &&
cd /tmp/LUSCA_HEAD-r14942 &&
./configure \
--prefix=/usr \
--exec_prefix=/usr \
--bindir=/usr/sbin \
--sbindir=/usr/sbin \
--libexecdir=/usr/lib/squid \
--sysconfdir=/etc/squid \
--localstatedir=/var/spool/squid \
--datadir=/usr/share/squid \
--enable-async-io=24 \
--with-aufs-threads=24 \
```

```

--with-pthreads \
--enable-storeio=aufs \
--enable-linux-netfilter \
--enable-arp-acl \
--enable-epoll \
--enable-removal-policies=heap \
--with-aio \
--with-dl \
--enable-snmp \
--enable-delay-pools \
--enable-htcp \
--enable-cache-digests \
--disable-unlinkd \
--enable-large-cache-files \
--with-large-files \
--enable-err-languages=English \
--enable-default-err-language=English \
--enable-referer-log \
--with-maxfd=65536 &&
make &&
make install

```

A continuación creamos el archivo de configuración /etc/squid.conf:

```
# nano /etc/squid/squid.conf
```

E introducimos este contenido (que ya es una modificación respecto del original de la web):

```

#####
## Squid_LUSCA configuration Starts from Here ...  #
## Thanks to Mr. Safatah [INDO] for sharing Configs  #
## Syed.Jahanzaib / 22nd April, 2014                #
## http://aacable.wordpress.com / aacable@hotmail.com #
#####

# HTTP Port for SQUID Service
http_port 192.168.1.4:8080 transparent
http_port 192.168.1.4:80 transparent
#http_port 127.0.0.1:3128 transparent
server_http11 on

# Cache Pee, for parent proxy if you have any, or ignore it.
#cache_peer x.x.x.x parent 8080 0

# Various Logs/files location
pid_filename /var/run/squid.pid
coredump_dir /var/spool/squid/
error_directory /usr/share/squid/errors/English
icon_directory /usr/share/squid/icons
mime_table /etc/squid/mime.conf
access_log daemon:/var/log/squid/access.log squid
cache_log /var/log/squid/cache.log
#debug_options ALL,1 22,3 11,2 #84,9

```

```

referer_log /var/log/squid/referer.log
cache_store_log /home/squid/var/log/squid/store.log
store_dir_select_algorithm round-robin
logfile_daemon /usr/lib/squid/logfile-daemon
logfile_rotate 1

# Cache Policy
cache_mem 64 MB
maximum_object_size_in_memory 0 KB
memory_replacement_policy heap GDSF
cache_replacement_policy heap LFUDA

minimum_object_size 0 KB
maximum_object_size 10 GB
cache_swap_low 98
cache_swap_high 99

# Cache Folder Path, using 5GB for test
cache_dir aufs /cache-1 5000 16 256

# ACL Section
acl all src all
acl manager proto cache_object
acl localhost src 127.0.0.1/32
acl to_localhost dst 127.0.0.0/8
acl localnet src 10.0.0.0/8      # RFC1918 possible internal network
acl localnet src 172.16.0.0/12   # RFC1918 possible internal network
acl localnet src 192.168.0.0/16  # RFC1918 possible internal network
acl localnet src 125.165.92.1    # RFC1918 possible internal network
acl SSL_ports port 443
acl Safe_ports port 80          # http
acl Safe_ports port 21          # ftp
acl Safe_ports port 443         # https
acl Safe_ports port 70          # gopher
acl Safe_ports port 210         # wais
acl Safe_ports port 1025-65535   # unregistered ports
acl Safe_ports port 280         # http-mgmt
acl Safe_ports port 488         # gss-http
acl Safe_ports port 591         # filemaker
acl Safe_ports port 777         # multiling http
acl CONNECT method CONNECT
acl purge method PURGE
acl snmppublic snmp_community public

acl range dstdomain .windowsupdate.com
range_offset_limit -1 KB range

#=====
# Loading Patch
acl DENYCACHE urlpath_regex \.(ini|ui|lst|inf|pak|ver|patch|md5|cfg|lst|list|src|log|conf|dbd|db)$
acl DENYCACHE urlpath_regex (notice.html|afs.dat|dat.asp|patchinfo.xml|version.list|iepngfix.htc|updates.txt|
patchlist.txt)
acl DENYCACHE urlpath_regex (pointblank.css|login_form.css|form.css|nouupdate.ui|ahn.ui|3n.mh)$
acl DENYCACHE urlpath_regex (Loader|gamenotice|sources|captcha|notice|reset)
no_cache deny DENYCACHE

range_offset_limit 1 MB !DENYCACHE
uri_whitespace strip

```

```
#=====
# Rules to block few Advertising sites
acl ads url_regex -i .youtube\.com\ad_frame?
acl ads url_regex -i .(s[s[0-90-9]])\.youtube\.com
acl ads url_regex -i .googlesyndication\.com
acl ads url_regex -i .doubleclick\.net
acl ads url_regex -i ^http://googleads\.
acl ads url_regex -i ^http://(ad|ads|ads[0-90-9]|ads\d|kad|a[b|d]|ad\d|adserver|adsbox)\.[a-z0-9]*\.[a-z][a-z]*
acl ads url_regex -i ^http://openx\.[a-z0-9]*\.[a-z][a-z]*
acl ads url_regex -i ^http://[a-z0-9]*\.openx\.net/
acl ads url_regex -i ^http://[a-z0-9]*\.u-ad\.info/
acl ads url_regex -i ^http://adserver\.bs/
acl ads url_regex -i !^http://adfl\.ly
http_access deny ads
http_reply_access deny ads
#deny_info http://yoursite/yourad,htm ads
#==== End Rules: Advertising ====

strip_query_terms off

acl yutub url_regex -i .*youtube\.com\.$
acl yutub url_regex -i .*youtu\.be\.$
logformat squid1 %{Referer}>h %ru
access_log /var/log/squid/yt.log squid1 yutub

# ==== Custom Option REWRITE ====
acl store_rewrite_list urlpath_regex \.(get_video\?|videodownload\?|videoplayback.*id)

acl store_rewrite_list urlpath_regex \.(mp2|mp3|mid|midi|mp[234]|wav|ram|ra|rm|au|3gp|m4r|m4a)\?
acl store_rewrite_list urlpath_regex \.(mpg|mpeg|mp4|m4v|mov|avi|asf|wmv|wma|dat|flv|swf)\?
acl store_rewrite_list urlpath_regex \.(jpeg|jpg|jpe|jp2|gif|tiff?|pcx|png|bmp|pic|ico)\?
acl store_rewrite_list urlpath_regex \.(chm|dll|doc|docx|xls|xlsx|ppt|pptx|pps|ppsx|mdb|mdbx)\?
acl store_rewrite_list urlpath_regex \.(txt|conf|cfm|psd|wmf|emf|vsd|pdf|rtf|odt)\?
acl store_rewrite_list urlpath_regex \.(class|jar|exe|gz|bz|bz2|tar|tgz|zip|gzip|arj|ace|bin|cab|msi|rar)\?
acl store_rewrite_list urlpath_regex \.(htm|html|mhtml|css|js)\?

acl store_rewrite_list_web url_regex ^http://([A-Za-z-]+[0-9]+)\.[A-Za-z]*\.[A-Za-z]*
acl store_rewrite_list_web_CDN url_regex ^http://[a-z]+[0-9]\.google\.com doubleclick\.net

acl store_rewrite_list_path urlpath_regex \.(mp2|mp3|mid|midi|mp[234]|wav|ram|ra|rm|au|3gp|m4r|m4a)$
acl store_rewrite_list_path urlpath_regex \.(mpg|mpeg|mp4|m4v|mov|avi|asf|wmv|wma|dat|flv|swf)$
acl store_rewrite_list_path urlpath_regex \.(jpeg|jpg|jpe|jp2|gif|tiff?|pcx|png|bmp|pic|ico)$
acl store_rewrite_list_path urlpath_regex \.(chm|dll|doc|docx|xls|xlsx|ppt|pptx|pps|ppsx|mdb|mdbx)$
acl store_rewrite_list_path urlpath_regex \.(txt|conf|cfm|psd|wmf|emf|vsd|pdf|rtf|odt)$
acl store_rewrite_list_path urlpath_regex \.(class|jar|exe|gz|bz|bz2|tar|tgz|zip|gzip|arj|ace|bin|cab|msi|rar)$
acl store_rewrite_list_path urlpath_regex \.(htm|html|mhtml|css|js)$

acl getmethod method GET

storeurl_access deny !getmethod
#this is not related to youtube video its only for CDN pictures
storeurl_access allow store_rewrite_list_web_CDN
storeurl_access allow store_rewrite_list_web store_rewrite_list_path
storeurl_access allow store_rewrite_list
storeurl_access deny all
storeurl_rewrite_program /etc/squid/storeurl.pl
storeurl_rewrite_children 10
```

```

storeurl_rewrite_concurrency 40
# ===== End Custom Option REWRITE =====

#=====
# Custom Option REFRESH PATTERN
#=====
refresh_pattern (get_video\?|videoplayback\?|videodownload\?|\.flv\?|\.fid\?) 43200 99% 43200 override-expire ignore-
reload ignore-must-revalidate ignore-private
refresh_pattern -i (get_video\?|videoplayback\?|videodownload\?) 5259487 999% 5259487 override-expire ignore-
reload reload-into-ims ignore-no-cache ignore-private
# -- refresh pattern for specific sites -- #
refresh_pattern ^http://*.jobstreet.com.*/* 720 100% 10080 override-expire override-lastmod ignore-no-cache
refresh_pattern ^http://*.indowebster.com.*/* 720 100% 10080 override-expire override-lastmod reload-into-ims
ignore-reload ignore-no-cache ignore-auth
refresh_pattern ^http://*.21cineplex.*/* 720 100% 10080 override-expire override-lastmod reload-into-ims ignore-
reload ignore-no-cache ignore-auth
refresh_pattern ^http://*.atmajaya.*/* 720 100% 10080 override-expire ignore-no-cache ignore-auth
refresh_pattern ^http://*.kompas.*/* 720 100% 10080 override-expire override-lastmod reload-into-ims ignore-no-
cache ignore-auth
refresh_pattern ^http://*.theinquirer.*/* 720 100% 10080 override-expire ignore-no-cache ignore-auth
refresh_pattern ^http://*.blogspot.com/* 720 100% 10080 override-expire override-lastmod reload-into-ims ignore-no-
cache ignore-auth
refresh_pattern ^http://*.wordpress.com/* 720 100% 10080 override-expire override-lastmod reload-into-ims ignore-
no-cache
refresh_pattern ^http://*.photobucket.com/* 720 100% 10080 override-expire override-lastmod reload-into-ims ignore-
no-cache ignore-auth
refresh_pattern ^http://*.tinypic.com/* 720 100% 10080 override-expire override-lastmod reload-into-ims ignore-no-
cache ignore-auth
refresh_pattern ^http://*.imageshack.us/* 720 100% 10080 override-expire override-lastmod reload-into-ims ignore-
no-cache ignore-auth
refresh_pattern ^http://*.kaskus.*/* 720 100% 28800 override-expire override-lastmod reload-into-ims ignore-no-cache
ignore-auth
refresh_pattern ^http://www.kaskus.com/* 720 100% 28800 override-expire override-lastmod reload-into-ims ignore-
no-cache ignore-auth
refresh_pattern ^http://*.detik.*/* 720 50% 2880 override-expire override-lastmod reload-into-ims ignore-no-cache
ignore-auth
refresh_pattern ^http://*.detiknews.*/* 720 50% 2880 override-expire override-lastmod reload-into-ims ignore-no-
cache ignore-auth
refresh_pattern ^http://video.liputan6.com/* 720 100% 10080 override-expire override-lastmod reload-into-ims ignore-
no-cache ignore-auth
refresh_pattern ^http://static.liputan6.com/* 720 100% 10080 override-expire override-lastmod reload-into-ims ignore-
no-cache ignore-auth
refresh_pattern ^http://*.friendster.com/* 720 100% 10080 override-expire override-lastmod ignore-no-cache ignore-
auth
refresh_pattern ^http://*.facebook.com/* 720 100% 10080 override-expire override-lastmod reload-into-ims ignore-no-
cache ignore-auth
refresh_pattern ^http://apps.facebook.com/* 720 100% 10080 override-expire override-lastmod reload-into-ims ignore-
no-cache ignore-auth
refresh_pattern ^http://*.fbcdn.net/* 720 100% 10080 override-expire override-lastmod reload-into-ims ignore-no-
cache ignore-auth
refresh_pattern ^http://profile.ak.fbcdn.net/* 720 100% 10080 override-expire override-lastmod reload-into-ims ignore-
no-cache ignore-auth
refresh_pattern ^http://static.playspoon.com/* 720 100% 10080 override-expire override-lastmod reload-into-ims
ignore-no-cache ignore-auth
refresh_pattern ^http://cooking.game.playspoon.com/* 720 100% 10080 override-expire override-lastmod reload-into-
ims ignore-no-cache ignore-auth
refresh_pattern -i http://[^a-z.]*onemanga.com/? 720 80% 10080 override-expire override-lastmod reload-into-ims
ignore-no-cache ignore-auth

```

```

refresh_pattern ^http://media?.onemanga.com/. * 720 80% 10080 override-expire override-lastmod reload-into-ims
ignore-no-cache ignore-auth
refresh_pattern ^http://*.yahoo.com/. * 720 80% 10080 override-expire override-lastmod reload-into-ims ignore-no-
cache ignore-auth
refresh_pattern ^http://*.google.com/. * 720 80% 10080 override-expire override-lastmod reload-into-ims ignore-no-
cache ignore-auth
refresh_pattern ^http://*.forummikrotik.com/. * 720 80% 10080 override-expire override-lastmod reload-into-ims
ignore-no-cache ignore-auth
refresh_pattern ^http://*.linux.or.id/. * 720 100% 10080 override-expire override-lastmod reload-into-ims ignore-no-
cache ignore-auth
# -- refresh pattern for extension -- #
refresh_pattern -i \.(mp2|mp3|mid|midi|mp[234]|wav|ram|ra|rm|au|3gp|m4r|m4a)(\?.*|$) 5259487 999% 5259487
override-expire ignore-reload reload-into-ims ignore-no-cache ignore-private
refresh_pattern -i \.(mpg|mpeg|mp4|m4v|mov|avi|asf|wmv|wma|dat|flv|swf)(\?.*|$) 5259487 999% 5259487 override-
expire ignore-reload reload-into-ims ignore-no-cache ignore-private
refresh_pattern -i \.(jpeg|jpg|jpe|jp2|gif|tiff?|pcx|png|bmp|pic|ico)(\?.*|$) 5259487 999% 5259487 override-expire
ignore-reload reload-into-ims ignore-no-cache ignore-private
refresh_pattern -i \.(chm|dll|doc|docx|xls|xlsx|ppt|pptx|pps|ppsx|mdb|mdbx)(\?.*|$) 5259487 999% 5259487 override-
expire ignore-reload reload-into-ims ignore-no-cache ignore-private
refresh_pattern -i \.(txt|conf|cfm|psd|wmf|emf|vsd|pdf|rtf|odt)(\?.*|$) 5259487 999% 5259487 override-expire ignore-
reload reload-into-ims ignore-no-cache ignore-private
refresh_pattern -i \.(class|jar|exe|gz|bz|bz2|tar|tgz|zip|gzip|arj|ace|bin|cab|msi|rar)(\?.*|$) 5259487 999% 5259487
override-expire ignore-reload reload-into-ims ignore-no-cache ignore-private
refresh_pattern -i \.(htm|html|mhtml|css|js)(\?.*|$) 1440 90% 86400 override-expire ignore-reload reload-into-ims
#=====
refresh_pattern -i (/cgi-bin/|\?) 0 0% 0
refresh_pattern ^gopher: 1440 0% 1440
refresh_pattern ^ftp: 10080 95% 10080 override-lastmod reload-into-ims
refresh_pattern . 0 20% 10080 override-lastmod reload-into-ims

http_access allow manager localhost
http_access deny manager
http_access allow purge localhost
#http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports

http_access allow localnet
http_access allow all
http_access deny all

icp_access allow localnet
icp_access deny all
icp_port 0

buffered_logs on

acl shoutcast rep_header X-HTTP09-First-Line ^ICY.[0-9]
upgrade_http0.9 deny shoutcast

acl apache rep_header Server ^Apache
broken_vary_encoding allow apache

forwarded_for off
header_access From deny all
header_access Server deny all
header_access Link deny all
header_access Via deny all
header_access X-Forwarded-For deny all

```

```
httpd_suppress_version_string on

shutdown_lifetime 10 seconds

snmp_port 3401
snmp_access allow snmppublic all
dns_timeout 1 minutes

dns_nameservers 8.8.8.8 8.8.4.4

fqdn_cache_size 5000 #16384
ipcache_size 5000 #16384
ipcache_low 98
ipcache_high 99
log_fqdn off
log_icp_queries off
memory_pools off

maximum_single_addr_tries 2
retry_on_error on

icp_hit_stale on

strip_query_terms off

query_icmp on
reload_into_ims on
emulate_httpd_log off
negative_ttl 0 seconds
pipeline_prefetch on
vary_ignore_expire on
half_closed_clients off
high_page_fault_warning 2
nonhierarchical_direct on
prefer_direct off
cache_mgr fj.detorres@gmail.com
cache_effective_user proxy
cache_effective_group proxy
visible_hostname strawberrylabs.
unique_hostname raspberry_pi
cachemgr_passwd none all
client_db on
max_filedescriptors 8192

# ZPH config Marking Cache Hit, so cached contents can be delivered at full lan speed via MT
zph_mode tos
zph_local 0x30
zph_parent 0
zph_option 136
```

Generamos ahora un archivo vacío que será nuestro script para ejecutar en perl  
/etc/squid/storeurl.pl

```
touch /etc/squid/storeurl.pl
chmod +x /etc/squid/storeurl.pl
nano /etc/squid/storeurl.pl
```



Y en éste introducimos:

```
#!/usr/bin/perl
#####
## Squid_LUSCA storeurl.pl starts from Here ...  #
## Thanks to Mr. Safatah [INDO] for sharing Configs #
## Syed.Jahanzaib / 22nd April, 2014 #
## http://aacable.wordpress.com / aacable@hotmail.com #
#####
$|=1;
while (<>) {
  @X = split;
  $x = $X[0] . " ";
  ##=====
  ## Encoding YOUTUBE
  ##=====
  if ($X[1] =~ m/^http:\V\.*(youtube|google).*videoplayback.*){
    @itag = m/[\&?](itag=[0-9]*)/;
    @CPN = m/[\&?]cpn\=[a-zA-Z0-9\-\_]*/;
    @IDS = m/[\&?]id\=[a-zA-Z0-9\-\_]*/;
    $id = &GetID($CPN[0], $IDS[0]);
    @range = m/[\&?](range=[^\&]*s*)/;
    print $x . "http://fathayu/" . $id . "&@itag@range\n";
  } elsif ($X[1] =~ m/(youtube|google).*videoplayback\?/ ) {
    @itag = m/[\&?](itag=[0-9]*)/;
    @id = m/[\&?](id=[^\&]*s*)/;
    @redirect = m/[\&?](redirect_counter=[^\&]*s*)/;
    print $x . "http://fathayu/";
  }
  # =====
  # VIMEO
  # =====
  } elsif ($X[1] =~ m/^http:\V\av\.vimeo\.com\Vd+\Vd+\V(.*)\?/ ) {
    print $x . "http://fathayu/" . $1 . "\n";
  } elsif ($X[1] =~ m/^http:\V\pdl\.vimeocdn\.com\Vd+\Vd+\V(.*)\?/ ) {
    print $x . "http://fathayu/" . $1 . "\n";
  }
  # =====
  # DAILYMOTION
  # =====
  } elsif ($X[1] =~ m/^http:\V\proxy-[0-9]{1}\.dailymotion\.com\V(.*)\V(.*)\Vvideo\Vd{3}\Vd{3}\V(.flv)/ ) {
    print $x . "http://fathayu/" . $1 . "\n";
  } elsif ($X[1] =~ m/^http:\V\vid[0-9]\.ak\.dmcdn\.net\V(.*)\V(.*)\Vvideo\Vd{3}\Vd{3}\V(.flv)/ ) {
    print $x . "http://fathayu/" . $1 . "\n";
  }
  # =====
  # YIMG
  # =====
  } elsif ($X[1] =~ m/^http:\V\(.*)\.yimg\.com\V(.*)\V([^\V?\&]*\V[^\V?\&]*\.[^\V?\&]{3,4})(\?.*)?\$/ ) {
    print $x . "http://fathayu/" . $3 . "\n";
  }
  # =====
  # YIMG DOUBLE
  # =====
  } elsif ($X[1] =~ m/^http:\V\(.*)\.yimg\.com\V(.*)\.yimg\.com\V(.*)\?(\.*)/ ) {
    print $x . "http://fathayu/" . $3 . "\n";
  }
  # =====
  # YIMG WITH &sig=
  # =====
  } elsif ($X[1] =~ m/^http:\V\(.*)\.yimg\.com\V(.*)/ ) {
    @y = ($1,$2);
    $y[0] =~ s/[a-z]+[0-9]+/cdn/;
```

26/55

27/55

```

}
if ($line =~ m/.*/youtube.*V.*[&?](video_id|docid|v)=([a-zA-Z0-9\-\_]*).*cpn=$CPN[0].*/){
$Id = $2;
last;
}
last if --$lim <= 0;
}
if ($Id eq ""){
$Id = $IDS[0];
}
$ref_log->close();
return $Id;
}
### STOREURL.PL ENDS HERE ###

```

Que es exactamente el original.

A continuación *squid* ha de generar el sistema de archivos de logs tal y como lo hemos configurado y el sistema de archivos de la cache (configurada para que sea /cache-1):

### squid -z #como root

Y lo lanzamos en modo depuración para observar posibles errores:

**sudo squid -d1N:**

```

pi@raspberrypi ~$ sudo squid -d1N
pi@raspberrypi ~$ 2014/06/15 22:55:45| Starting Squid Cache version LUSCA_HEAD-r
14942 for armv6l-unknown-linux-gnu...
2014/06/15 22:55:45| Process ID 21333
2014/06/15 22:55:45| NOTICE: Could not increase the number of filedescriptors
2014/06/15 22:55:45| With 1024 file descriptors available
2014/06/15 22:55:45| Using epoll for the IO loop
2014/06/15 22:55:45| Performing DNS Tests...
2014/06/15 22:55:45| Successful DNS name lookup tests...
2014/06/15 22:55:45| Adding nameserver 8.8.8.8 from squid.conf
2014/06/15 22:55:45| Adding nameserver 8.8.4.4 from squid.conf
2014/06/15 22:55:45| helperOpenServers: Starting 10 'storeurl.pl' processes
2014/06/15 22:55:45| logfileOpen: opening log /var/log/squid/referer.log
2014/06/15 22:55:45| logfileOpen: opening log daemon:/var/log/squid/access.log
2014/06/15 22:55:45| Logfile Daemon: opening log /var/log/squid/access.log
2014/06/15 22:55:45| logfileOpen: opening log /var/log/squid/yt.log
2014/06/15 22:55:45| Swap maxSize 5120000 + 65536 KB, estimated 398887 objects
2014/06/15 22:55:45| Target number of buckets: 19944
2014/06/15 22:55:45| Using 32768 Store buckets
2014/06/15 22:55:45| Max Mem size: 65536 KB
2014/06/15 22:55:45| Max Swap size: 5120000 KB
2014/06/15 22:55:45| Local cache digest enabled; rebuild/rewrite every 3600/3600
sec
2014/06/15 22:55:45| logfileOpen: opening log /home/squid/var/log/squid/store.log
2014/06/15 22:55:45| AUFS: /cache-1: log '/cache-1/swap.state' opened on FD 23
2014/06/15 22:55:45| AUFS: /cache-1: tmp log /cache-1/swap.state.new opened on FD
23
2014/06/15 22:55:45| Rebuilding storage in /cache-1 (CLEAN)
2014/06/15 22:55:45| Using Round Robin store dir selection
2014/06/15 22:55:45| Set Current Directory to /var/spool/squid/
2014/06/15 22:55:45| Loaded Icons.
2014/06/15 22:55:47| Accepting transparently proxied HTTP connections at 192.168.
1.4, port 8080, FD 25.
2014/06/15 22:55:47| Accepting transparently proxied HTTP connections at 192.168.
1.4, port 80, FD 26.
2014/06/15 22:55:47| Accepting HTCP messages on port 4827, FD 27.
2014/06/15 22:55:47| Accepting SNMP messages on port 3401, FD 28.
2014/06/15 22:55:47| WCCP Disabled.
2014/06/15 22:55:47| Ready to serve requests.
2014/06/15 22:55:49| Store rebuilding is 100.0% complete
2014/06/15 22:55:49| /cache-1: completed rebuild
2014/06/15 22:55:49| Done scanning /cache-1 (2418 entries)
2014/06/15 22:55:49| AUFS: /cache-1: tmp log closed on FD 23
2014/06/15 22:55:49| AUFS: /cache-1: post-rename; log /cache-1/swap.state, opened
on FD 23
2014/06/15 22:55:49| Finished rebuilding storage from disk.
2014/06/15 22:55:49| 2418 Entries scanned
2014/06/15 22:55:49| 0 Invalid entries.
2014/06/15 22:55:49| 0 With invalid flags.
2014/06/15 22:55:49| 2418 Objects loaded.
2014/06/15 22:55:49| 0 Objects expired.
2014/06/15 22:55:49| 0 Objects cancelled.
2014/06/15 22:55:49| 0 Duplicate URLs purged.
2014/06/15 22:55:49| 0 Swapfile clashes avoided.
2014/06/15 22:55:49| Took 3.4 seconds ( 719.1 objects/sec).
2014/06/15 22:55:49| Beginning Validation Procedure
2014/06/15 22:55:49| Completed Validation Procedure
2014/06/15 22:55:49| Validated 2418 Entries
2014/06/15 22:55:49| store_swap_size = 657020k
2014/06/15 22:55:49| storeLateRelease: released 0 objects
pi@raspberrypi ~$

```

Comprobamos que sigue en ejecución:

```
pi@raspberrypi ~ $ ps aux | grep squid
root      7275  0.0  0.2  6480 1124 ?        Ss   22:41   0:00 squid -d1N
proxy     7277 12.3  1.4 16204 6964 ?        Sl   22:41   0:32 (squid) -d1N
proxy     7281  0.1  0.5  4868 2624 ?        Ss   22:41   0:00 /usr/bin/perl /etc/squid/storeurl.pl
proxy     7282  0.1  0.5  4868 2624 ?        Ss   22:41   0:00 /usr/bin/perl /etc/squid/storeurl.pl
proxy     7283  0.1  0.5  4868 2584 ?        Ss   22:41   0:00 /usr/bin/perl /etc/squid/storeurl.pl
proxy     7284  0.1  0.5  4868 2584 ?        Ss   22:41   0:00 /usr/bin/perl /etc/squid/storeurl.pl
proxy     7285  0.1  0.5  4868 2584 ?        Ss   22:41   0:00 /usr/bin/perl /etc/squid/storeurl.pl
proxy     7286  0.1  0.5  4868 2584 ?        Ss   22:41   0:00 /usr/bin/perl /etc/squid/storeurl.pl
proxy     7289  0.1  0.5  4868 2584 ?        Ss   22:41   0:00 /usr/bin/perl /etc/squid/storeurl.pl
proxy     7291  0.1  0.5  4868 2584 ?        Ss   22:41   0:00 /usr/bin/perl /etc/squid/storeurl.pl
proxy     7292  0.1  0.5  4868 2584 ?        Ss   22:41   0:00 /usr/bin/perl /etc/squid/storeurl.pl
proxy     7294  0.1  0.5  4868 2584 ?        Ss   22:41   0:00 /usr/bin/perl /etc/squid/storeurl.pl
proxy     7301  0.0  0.1  2472   568 ?        Ss   22:41   0:00 (logfile-daemon) /var/log/squid/access.log
pi        11733 0.0  0.1  2020   632 pts/0    S+   22:46   0:00 grep --color=auto squid
pi@raspberrypi ~ $
```

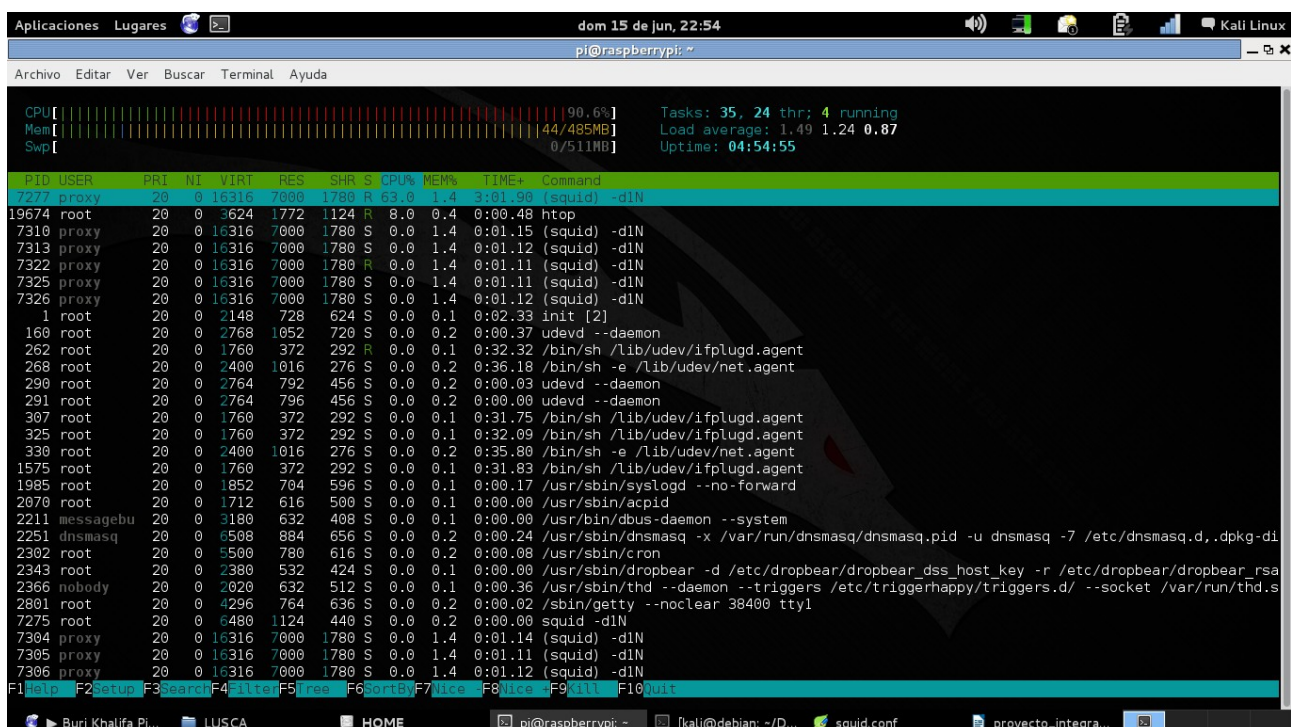
Podemos hacer una prueba de cacheo de un vídeo en resolución 4k (ultra HD).

Hay que aclarar que un *proxy caché* no puede “cachear” una comunicación *https* porque esta va cifrada y para eso tendría que suplantar el certificado digital de la web que ofrece la conexión segura por *https*. Así pues, para “cachear” vídeos de youtube y contenido en general, debemos asegurarnos de que nuestra conexión se realiza a <http://www.youtube.com>

Vemos, por ejemplo, el siguiente vídeo en 4k:

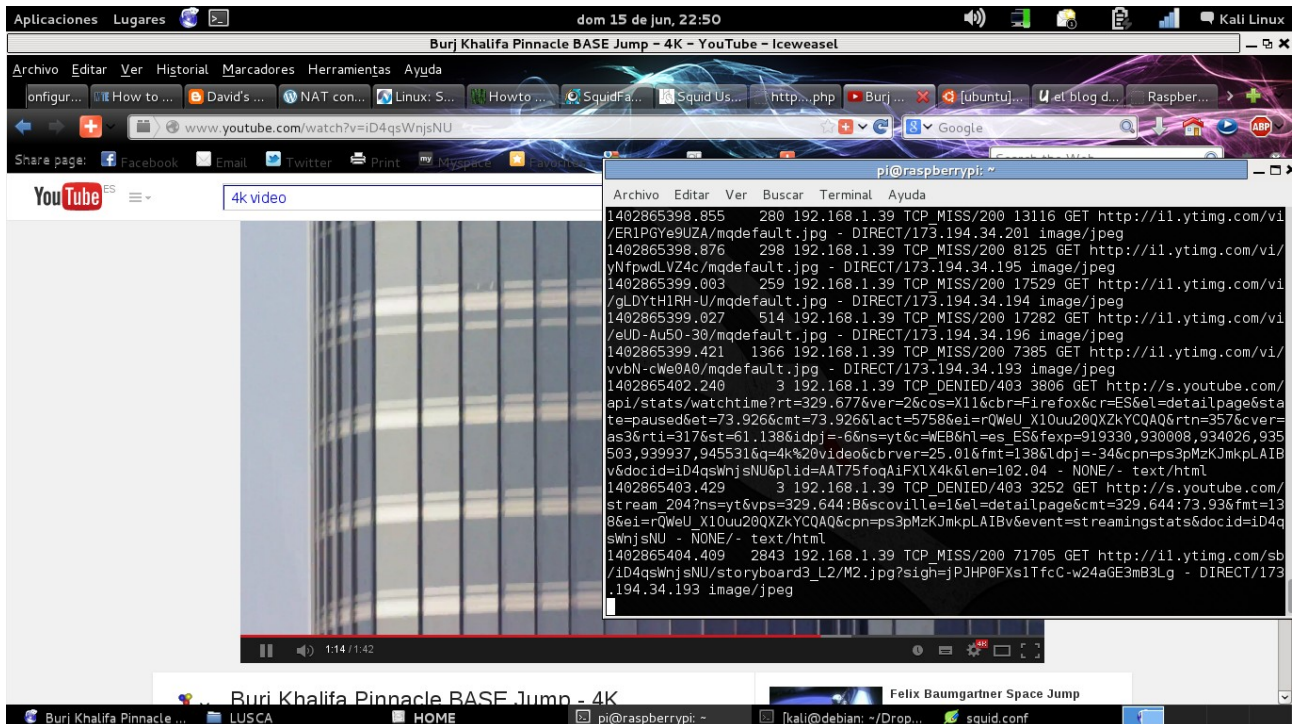
<http://www.youtube.com/watch?v=iD4qsWnjsNU>

Miramos el rendimiento del sistema a través del comando *htop*:

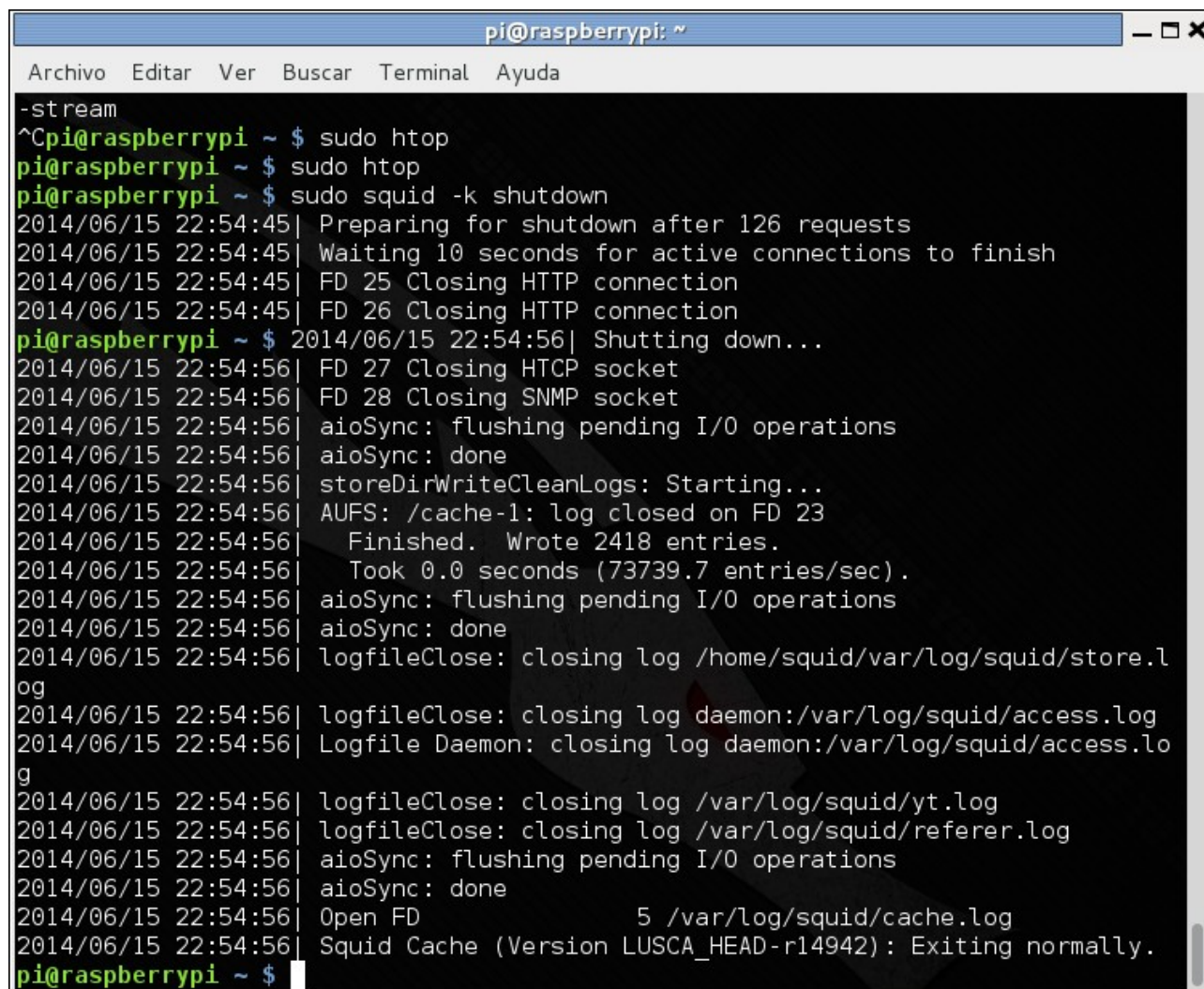


Mientras se está viendo el vídeo, podemos también ver el log de acceso en tiempo real según va modificándose con el comando **sudo tailf /var/log/squid/access.log**:





Procedemos también a apagar *squid* con el comando `sudo squid -k shutdown` que esperará 10 segundos a que terminen de cerrarse todas las conexiones:



```
pi@raspberrypi: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
-stream  
^Cpi@raspberrypi ~ $ sudo htop  
pi@raspberrypi ~ $ sudo htop  
pi@raspberrypi ~ $ sudo squid -k shutdown  
2014/06/15 22:54:45| Preparing for shutdown after 126 requests  
2014/06/15 22:54:45| Waiting 10 seconds for active connections to finish  
2014/06/15 22:54:45| FD 25 Closing HTTP connection  
2014/06/15 22:54:45| FD 26 Closing HTTP connection  
pi@raspberrypi ~ $ 2014/06/15 22:54:56| Shutting down...  
2014/06/15 22:54:56| FD 27 Closing HTCP socket  
2014/06/15 22:54:56| FD 28 Closing SNMP socket  
2014/06/15 22:54:56| aioSync: flushing pending I/O operations  
2014/06/15 22:54:56| aioSync: done  
2014/06/15 22:54:56| storeDirWriteCleanLogs: Starting...  
2014/06/15 22:54:56| AUFS: /cache-1: log closed on FD 23  
2014/06/15 22:54:56| Finished. Wrote 2418 entries.  
2014/06/15 22:54:56| Took 0.0 seconds (73739.7 entries/sec).  
2014/06/15 22:54:56| aioSync: flushing pending I/O operations  
2014/06/15 22:54:56| aioSync: done  
2014/06/15 22:54:56| logfileClose: closing log /home/squid/var/log/squid/store.l  
og  
2014/06/15 22:54:56| logfileClose: closing log daemon:/var/log/squid/access.log  
2014/06/15 22:54:56| Logfile Daemon: closing log daemon:/var/log/squid/access.lo  
g  
2014/06/15 22:54:56| logfileClose: closing log /var/log/squid/yt.log  
2014/06/15 22:54:56| logfileClose: closing log /var/log/squid/referer.log  
2014/06/15 22:54:56| aioSync: flushing pending I/O operations  
2014/06/15 22:54:56| aioSync: done  
2014/06/15 22:54:56| Open FD 5 /var/log/squid/cache.log  
2014/06/15 22:54:56| Squid Cache (Version LUSCA_HEAD-r14942): Exiting normally.  
pi@raspberrypi ~ $
```

Prueba de “cacheo” de paquetes apt:

En este vídeo subido por mí se puede ver el efecto el *proxy squid* en el almacenamiento en cache, de archivos descargados a través del gestor inteligente de paquetes *apt* cuando actualizamos una máquina virtual *Kali Linux* y luego hacemos lo mismo con la máquina real *host* desde la que estamos ejecutando la MV. Como parte de los paquetes a actualizar son los mismos (*metaexploit*, concretamente), se puede observar cómo, en la segunda descarga (cuando la máquina física procede a actualizarse, un aumento de la velocidad de descarga desde un máximo de algo más de 600 KB/s a más o menos 2,5 MB/s. Teniendo en cuenta que el ancho de banda con el que contamos para las pruebas es de 6 Mbits/s teóricos, es imposible obtener una velocidad de descarga de 2,5 MB/s si no es porque los paquetes están siendo descargados desde la red local.

[https://www.youtube.com/watch?v=uAz\\_02DWl8c&feature=youtu.be](https://www.youtube.com/watch?v=uAz_02DWl8c&feature=youtu.be)

Script de automatización del proceso de instalación de *lusca/squid* en *Raspberrypi* (aporte personal):

```
#!/bin/bash
#Este script tiene en cuenta que squid ya esté instalado tanto con aptitude
#como con este stript u otra manera de compilación desde su código fuente:

#Eliminamos instalaciones anteriores:
if [[ $(dpkg -l | grep "squid") != "" ]]
then
    killall squid
    apt-get remove --purge squid
    apt-get remove --purge squid3
    apt-get autoremove
    rm -fr /etc/squid
    rm -fr /var/log/squid
fi

if [[ -d /cache-1 ]]
then
    rm -fr /cache-1
fi

#Comienza la instalación de los paquetes necesarios para compilar squid
# su fork "lusca" desde el código fuente. Obsérvese el uso de "&&" puesto
#que necesitamos cada una de las herramientas para que funcionen las posteriores:

apt-get update &&
apt-get install gcc -y &&
apt-get install build-essential -y &&
apt-get install libstdc++6 -y &&
apt-get install unzip -y &&
apt-get install bzip2 -y &&
apt-get install sharutils -y &&
apt-get install ccze -y &&
apt-get install libzip-dev -y &&
apt-get install automake1.9 -y &&
apt-get install acpid -y &&
apt-get install libfile-readbackwards-perl -y &&
apt-get install dnsmasq -y &&

#controlamos el error de directorio existente enviado el error a /dev/null en vez de usar un condicional como en los
casos anteriores:
mkdir /lusca && 2>/dev/null
#Este es nuestro directorio de trabajo. Las instrucciones originales utilizan /tmp
#pero yo veo útil preservar la compilación para poder desinstalar squid en
#caso necesario:
cd /lusca &&
wget -c http://wifismartzone.com/files/linux_related/lusca/LUSCA_HEAD-r14942.tar.gz &&
tar -xvzf LUSCA_HEAD-r14942.tar.gz &&
cd /lusca/LUSCA_HEAD-r14942 &&
./configure \
--prefix=/usr \
--exec_prefix=/usr \
--bindir=/usr/sbin \
--sbindir=/usr/sbin \
--libexecdir=/usr/lib/squid \
--sysconfdir=/etc/squid \
--localstatedir=/var/spool/squid \
--datadir=/usr/share/squid \
```



```
--enable-async-io=24 \
--with-aufs-threads=24 \
--with-pthreads \
--enable-storeio=aufs \
--enable-linux-netfilter \
--enable-arp-acl \
--enable-epoll \
--enable-removal-policies=heap \
--with-aio \
--with-dl \
--enable-snmp \
--enable-delay-pools \
--enable-htcp \
--enable-cache-digests \
--disable-unlinkd \
--enable-large-cache-files \
--with-large-files \
--enable-err-languages=English \
--enable-default-err-language=English \
--enable-referer-log \
--with-maxfd=65536 &&
make &&
make install
```

```
cd /etc/squid/
```

#Bajo de mi dropbox los archivos de configuración necesarios para que funcione squid de la forma que #perseguiamos:

```
wget https://www.dropbox.com/s/zzzo1g9f3tybm15/squid.conf
wget https://www.dropbox.com/s/qbzbs03hhr3a4rl/storeurl.pl
```

```
chmod +x /etc/squid/storeurl.pl
```

# Carpeta de logueo y asignación de permisos:  
# Mandamos los errores a /dev/null si ya existe:

```
mkdir /var/log/squid 2>/dev/null
chown proxy:proxy /var/log/squid/
```

#Creamos una segunda ruta de archivos de logueo  
mkdir -p /home/squid/var/log/squid/  
chown proxy:proxy /home/squid

# Cache Folder  
mkdir /cache-1 2>/dev/null  
chown proxy:proxy /cache-1

#Generación de los directorios de cache de squid:  
squid -z

#Lanzamos squid:  
squid

#Comprobamos si se está ejecutando correctamente. Tienen que verse  
#múltiples instancias de squid ejecutando el script store.pl (en perl):  
ps aux | grep squid

**La configuración de Lusca/Squid con su archivo de configuración /etc/squid/squid.conf:**

Puerto de escucha de squid: se eligen dos. Por un lado, el puerto 80, imprescindible para que actúe como proxy transparente, puesto que lo que se pretende es que los navegadores no tengan que ser configurados al efecto. Para este propósito además, deberemos redirigir también el tráfico a través de un enrutador como *iptables*, como se verá después.

El puerto 8080 se incluye también (puede ponerse más de un puerto a la escucha). La idea es que muchos navegadores, de estar ya configurados (porque así lo haya establecido el administrador del sistema de alguna red), para conectarse a través de un proxy, el puerto que suele usarse es el puerto 8080. Si además queremos asegurarnos de que, en caso de que algún navegador haya sido configurado también para enviar las peticiones a un proxy squid típicamente configurado, podríamos añadir además la escucha al puerto 3128, que es el que por defecto tiene configurado squid:

```
# HTTP Port for SQUID Service
http_port 192.168.1.4:8080 transparent
http_port 192.168.1.4:80 transparent
http_port 192.168.1.4:3128 transparent
```

Especificamos la ip del proxy porque se han obtenido ciertos errores en la entrega del contenido en cache parecen resolverse de esta forma, así como haber comentado una regla del grupo `http_access`. Eso redundaría en un mejor rendimiento del visionado de contenido en streaming de sitios como youtube.

```
server_http11 on
```

Permite que se transfiera la petición del protocolo *http* **versión 1.1** de los navegadores que expresamente la realicen. Resuelve un fallo de una mala implementación (quizás ya corregida en la actualidad) de la petición expresa de la versión 1.1 del protocolo http en la cabecera de respuesta de páginas como facebook y google, según puede apreciarse [aquí](#).

*Squid* es un *proxy* que puede actuar como “hijo” de otro principal o como “hermano” o *sibling*, caso en el que habría un balanceo de carga entre todos ellos (para redes grandes). En ambos casos lo que se le dice al *proxy* es que tenga en cuenta el contenido que ya pueden tener los otros almacenados en su cache para no volver a descargarlo.

La siguiente línea comentada para que no tenga efecto, salvo que se necesite, informa a nuestro servidor *squid* de que hay un proxy *padre* en la ip *x.x.x.x* escuchando las peticiones http en el puerto 8080, y 0 corresponde al puerto a la escucha del protocolo ICP, que es el que transfiere peticiones de caché ([véase esta entrada de internet](#)). Cuando el puerto se ha establecido con un valor 0 (cero) es que ha sido deshabilitado según puede verse más adelante:

```
# Cache Pee, for parent proxy if you have any, or ignore it.
#cache_peer x.x.x.x parent 8080 0
```

Los distintos lugares donde se almacenan los registros de actividad de *squid*:

```
# Various Logs/files location
pid_filename /var/run/squid.pid
```

Archivo donde squid guarda el *pid* del proceso con el que ha sido lanzado. Esta es la razón por la

que nuestro script de inicio `/etc/ini.d/capture_connection.sh` puede contener una orden que lanza squid sin comprobar si este ya está en ejecución. Precisamente *squid* comprueba cuando es ejecutado, si este fichero existe y contiene el *pid* de proceso en ejecución de squid. En caso de que ya exista significa que *squid* se está ejecutando y no volverá a lanzarse una instancia nueva del mismo.

*coredump\_dir* `/var/spool/squid/`

Se refiere al lugar en el que squid hará un volcado de memoria (para posterior depuración de errores) en caso de que falle. Se suele elegir una partición distinta para esto menesteres porque el volcado podría ser muy grande y dejar sin memoria al resto de la partición en la que squid trabaja y genera la caché.

*error\_directory* `/usr/share/squid/errors/English`

Lugar en el que se almacenan páginas web básicas que muestran mensajes de error al usuario (como que un contenido ha sido bloqueado por squid, etc). Por defecto se instala “English”, es decir, conjunto de mensajes en inglés.

*icon\_directory* `/usr/share/squid/icons`

Los iconos utilizados por squid para mostrar listados de servidores FTP y GOPHER.

*mime\_table* `/etc/squid/mime.conf`

El contenido mime ha de ser usado por squid cuando recibe una respuesta de un servidor FTP o GOPHER que es vista a través de un cliente http, y ser incluido en esta, puesto que es lo que esperan los navegadores web y dichos servidores no la sirven; por ser este elemento propio del protocolo http, no así de los protocolos FTP y GOPHER:

*access\_log daemon:*`/var/log/squid/access.log squid`

Es el archivo de logs de conexiones de squid. Nos interesa saber que en las líneas donde aparece:

MISS: el contenido es servido directamente desde la conexión a internet

HIT: el contenido es servido por squid desde su cache.

*cache\_log* none

*#debug\_options* ALL,1 22,3 11,2 #84,9

*referer\_log* `/var/log/squid/referer.log`

*cache\_store\_log* `/home/squid/var/log/squid/store.log`

Cuando hemos incluido esta línea anterior, que el archivo original de configuración de la variante de squid llamada *lusca* tenía configurada para no hacer *logeo* se ha obtenido un error que origina la detección de *squid* según es lanzado. Termina con un “error fatal” porque no puede escribir sobre el archivo, que por defecto suele ser `/var/log/squid/store.log` porque no encuentra permisos de acceso

para *proxy:proxy* pese a que todo eso está correctamente configurado para el resto de los archivos de log. Ha ocurrido que ese cambio se ha realizado en el momento de escribir esta documentación y antes no estaba implementada. El error era exactamente éste:

*Cannot open '/var/log/store.log' for writing. The parent directory must be writable by the user 'proxy', which is the cache\_effective\_user set in squid.conf.*

Borrando los archivos creados específicamente para el uso de *squid*, que son */var/log/squid* y, en nuestro caso, */cache-1* y relanzando el comando *squid -z* para que estos vuelvan a ser generados, no resuelve el problema. Se ha optado finalmente por crear el archivo */home/squid/var/log/squid/* y asignarle el dueño y grupo *proxy* puesto que es éste el usuario definido en *squid.conf*. Otras soluciones propuestas en los foros pasa por asignar permisos totales al directorio de logs de *squid*, lo que resulta, desde mi punto de vista, demasiado inseguro. De todas formas, *store.log* simplemente almacena de forma temporal los sucesos relativos al funcionamiento de *squid*. Es ahí donde podemos observar qué errores han ocurrido en un funcionamiento anormal de éste.

*store\_dir\_select\_algorithm round-robin*

Algoritmo utilizado para elegir en qué directorios guarda qué clase de objetos a ser “cacheados”.

*logfile\_daemon /usr/lib/squid/logfile-daemon*

*logfile\_rotate 1*

Rotación de archivos de log activa. Sirve para que sea efectiva la orden *squid -k rotate* y se genere un nuevo archivo de logs. De esa manera podemos destruir los antiguos, o comprimirlos y eliminarlos del directorio y así ahorrar espacio en disco. El paquete *logrotate*, que no ha sido implementado en esta ocasión (hay intención de seguir madurando este proyecto para uso personal más allá del “proyecto integrado”), realizaría de forma programada y automática esa rotación de logs. Este proceso de rotación de logs es habitual en otros procesos que generan archivos de informes en linux.

*# Cache Policy*  
*cache\_mem 64 MB*

Cache de almacenamiento en memoria de los objetos en tránsito. Es decir, aquellos que van a ser copiados en el directorio de *cache* de *squid*. Se considera el tercer factor de consumo total de memoria RAM por parte del servidor *squid*.

*maximum\_object\_size\_in\_memory 0 KB*

Establecido a cero implica “sin límite de tamaño”.

*cache\_replacement\_policy heap LFUDA*

¿Con qué criterio se borrar objetos existentes en caché para poder ser reemplazados por objetos nuevos cuando no hay espacio suficiente en disco? Refiriéndonos, lógicamente, al espacio máximo asignado para el “cacheo” de objetos, este algoritmo preserva los objetos “más populares” frente a

los menos, sin importar la fecha de “cacheo”. Tampoco le afecta el tamaño del archivo como criterio para ser borrado (archivos más pesados no serán borrados primero si estos son realmente más relevantes o “populares”). Por ejemplo: un vídeo de streaming que veamos muy a menudo, aunque tenga un “peso” elevado, permanecerá más tiempo que cualquier otro contenido más “ligero” al que se acceda mucho menos.

*memory\_replacement\_policy heap GDSF*

Algoritmo para seleccionar qué objetos serán borrados de la zona de memoria reservada en RAM para la ejecución de squid. El algoritmo GDSF da prioridad a los objetos más frecuentemente descargados teniendo prioridad los de menor tamaño frente a los “pesados”. Frecuencia de acceso y poco tamaño son los dos factores que le dan prioridad frente a otros a la hora de seleccionar cuál ha de ser borrado si la memoria asignada (en este caso, RAM) comienza a escasear y se hace necesario borrar objetos.

*minimum\_object\_size 0 KB*

Tamaño mínimo del objeto para ser almacenado. Cero significa sin tamaño mínimo.

*maximum\_object\_size 10 GB*

Por analogía con lo anterior, tamaño máximo que puede tener un objeto. Tamaño máximo de un archivo. Diez gigabytes en este caso.

*cache\_swap\_low 98*

*cache\_swap\_high 99*

Límites mínimo y máximo de espacio ocupado en disco (de entre la cantidad asignada) a partir del cual comienzan a borrarse objetos de la caché. ¿En qué momento squid “decide” que ya es momento de borrar objetos? No es necesariamente cuando la capacidad de almacenamiento ha sido ocupada al 100. Los límites mínimo y máximo establecen, **en porcentajes**, la cantidad de espacio que ha de haberse ocupado para comenzar el borrado de objetos siguiendo el algoritmo elegido en la variable *cache\_replacement\_policy*. *Cache\_swap\_low* establece la cantidad de espacio máximo que puede ser ocupado, y a partir del cual ha de iniciarse el borrado. El límite máximo *cache\_swap\_high* establece el límite que no puede, bajo ningún concepto, ser rebasado. En este caso, la prioridad deja de ser cumplir el algoritmo de selección de objetos. Ahora la prioridad es borrar.

El espacio asignado en el disco duro para el almacenamiento de objetos es el que se indica con la variable a continuación:

*# Cache Folder Path, using 5GB for test*

*cache\_dir aufs /cache-1 5000 16 256*

Que significa que se han dedicado 5Gb de espacio en disco en la ruta indicada (*/cache-1*). Los números que vienen a continuación significa que serán divididos en hasta 16 directorios con un máximo de hasta 256 niveles de subdirectorios en los que guardar en caché.

Nota importante: Si asignamos un espacio de almacenamiento mayor del espacio libre real que tenemos en el disco, eso *squid* no lo detectará y el programa directamente “se colgará”.

Sección de las *ACL*'s o reglas de acceso:

*# ACL Section*

*acl all src all*

*acl manager proto cache\_object* #definimos el acl “manager” para establecer qué hacemos con el protocolo *cache\_object*, que se utiliza para pedir y entregar contenido en cache ( conexión telnet o ssh, método GET):

telnet mycache.example.com 3128

GET cache\_object://mycache.example.com/info HTTP/1.0

Como se indica más abajo, nuestra regla será que sólo tenga acceso el propio servidor (localhost). Denegado para el resto. Esto es, ninguna máquina puede solicitar el contenido en caché a nuestro servidor.

*acl localhost src 127.0.0.1/32* #definimos qué direcciones IP **de destino** entendemos por “localhost” para luego aplicarles reglas. Ídem para el resto de las entradas en las que se definen listas *acl* como *src* (source):

*acl to\_localhost dst 127.0.0.0/8*

*acl localnet src 10.0.0.0/8* # RFC1918 possible internal network

*acl localnet src 172.16.0.0/12* # RFC1918 possible internal network

*acl localnet src 192.168.0.0/16* # RFC1918 possible internal network

*acl localnet src 125.165.92.1* # RFC1918 possible internal network

En estas *acl* estamos definiendo las típicas IP privadas con las que nos podemos encontrar, con la intención de definir, más adelante que hácer con todas ellas. Como se verá, se les permitirá la conexión al proxy, puesto que la idea es que los clientes locales accedan a la *cache* del proxy y ahorrar ancho de banda.

Asignación de un conjunto de puertos a la *acl* “Safe\_ports” y asignación del puerto 443 a “SSL\_ports”:

*acl SSL\_ports port 443*

*acl Safe\_ports port 80* # http

*acl Safe\_ports port 21* # ftp

*acl Safe\_ports port 443* # https

*acl Safe\_ports port 70* # gopher

*acl Safe\_ports port 210* # wais

*acl Safe\_ports port 1025-65535* # unregistered ports

*acl Safe\_ports port 280* # http-mgmt

*acl Safe\_ports port 488* # gss-http

*acl Safe\_ports port 591* # filemaker

*acl Safe\_ports port 777* # multiling http

*acl CONNECT method CONNECT*

ACL llamada *connect* que especifica el método de conexión. Connect significa que el proxy hará de túnel entre el cliente (hace una petición GET o POST) al servidor y su respuesta. Para una conexión https, como el proxy no puede descifrar la conexión segura entre el cliente y el servidor, simplemente abre un túnel entre ambos, esto es, permite la conexión directa.

(<http://stackoverflow.com/questions/11697943/when-should-one-use-connect-and-get-http-methods-at-http-proxy-server>)

acl purge method PURGE

Este “método” lo usa el programa squidclient para realizar consultas de tipo estadístico sobre el uso (y posibles fallos) ocurridos en squid. Es decir, nos permite que un programa instalado en local, como puede ser squidclient tenga acceso a esos datos para elaborar estadísticas:

acl snmppublic snmp\_community public

acl range dstdomain .windowsupdate.com  
range\_offset\_limit -1 KB range

Esta acl tiene que ver con el tamaño del archivo a partir del cual squid envía la petición al cliente después de haberlo “cacheado” o, en caso de sobrepasar dicho tamaño, opta por dejar pasar el paquete directamente sin “cacheo” previo.

Ejemplo de petición **squidclient -p 8080 mgr:info:**

HTTP/1.0 200 OK

Date: Sun, 15 Jun 2014 09:43:56 GMT

Content-Type: text/plain

Expires: Sun, 15 Jun 2014 09:43:56 GMT

X-Cache: MISS from proxy.zaib

X-Cache-Lookup: MISS from proxy.zaib:8080

Connection: close

Squid Object Cache: Version LUSCA\_HEAD-r14942

Start Time: Sun, 15 Jun 2014 08:21:20 GMT

Current Time: Sun, 15 Jun 2014 09:43:56 GMT

Connection information for squid:

Number of clients accessing cache: 1  
Number of HTTP requests received: 2  
Number of ICP messages received: 0  
Number of ICP messages sent: 0  
Number of queued ICP replies: 0  
Number of HTCP messages received: 0  
Number of HTCP messages sent: 0  
Request failure ratio: 0.00  
Average HTTP requests per minute since start: 0.0  
Average ICP messages per minute since start: 0.0  
Select loop called: 20531 times, 241.397 ms avg

Cache information for squid:

Request Hit Ratios: 5min: 0.0%, 60min: 0.0%  
Byte Hit Ratios: 5min: 100.0%, 60min: 100.0%  
Request Memory Hit Ratios: 5min: 0.0%, 60min: 0.0%  
Request Disk Hit Ratios: 5min: 0.0%, 60min: 0.0%  
Storage Swap size: 1878384 KB  
Storage Mem size: 196 KB  
Mean Object Size: 275.91 KB  
Requests given to unlinkd: 0

Median Service Times (seconds) 5 min 60 min:

HTTP Requests (All): 0.00000 0.00000  
Cache Misses: 0.00000 0.00000  
Cache Hits: 0.00000 0.00000  
Near Hits: 0.00000 0.00000

```

Not-Modified Replies: 0.00000 0.00000
DNS Lookups:         0.00000 0.00000
ICP Queries:         0.00000 0.00000
Resource usage for squid:
  UP Time:           4956.116 seconds
  CPU Time:          0.652 seconds
  CPU Usage:         0.01%
  CPU Usage, 5 minute avg: 0.01%
  CPU Usage, 60 minute avg: 0.01%
  Process Data Segment Size via sbrk(): 4280 KB
  Maximum Resident Size: 30160 KB
  Page faults with physical i/o: 1
Memory usage for squid via mallinfo():
  Total space in arena: 4280 KB
  Ordinary blocks:     4216 KB   13 blks
  Small blocks:        0 KB     0 blks
  Holding blocks:      1144 KB   2 blks
  Free Small blocks:   0 KB
  Free Ordinary blocks: 63 KB
  Total in use:        5360 KB 99%
  Total free:          63 KB 1%
  Total size:          5424 KB
Memory accounted for:
  Total accounted:     1033 KB
  memPoolAlloc calls: 50747
  memPoolFree calls: 36266
File descriptor usage for squid:
  Maximum number of file descriptors: 1024
  Largest file desc currently in use: 25
  Number of file desc currently in use: 23
  Files queued for open: 0
  Available number of file descriptors: 1001
  Reserved number of file descriptors: 100
  Store Disk files open: 0
  IO loop method:      epoll
Internal Data Structures:
  6835 StoreEntries
  27 StoreEntries with MemObjects
  26 Hot Object Cache Items
  6808 on-disk objects

```

```

#=====
=====

```

Reglas de acceso que impiden el “cacheo” de cierto tipo de archivos según su extension. A través del uso de expresiones regulares. Nótese que el “.” ha sido “escapado” para ser usado como carácter porque tiene significado propio (coincidencia de una o ninguna vez de la regla anterior a él):

#### # Loading Patch

```

acl DENYCACHE urlpath_regex \.(ini|ui|lst|inf|pak|ver|patch|md5|cfg|lst|list|rsc|log|conf|dbd|db)$
acl DENYCACHE urlpath_regex (notice.html|afs.dat|dat.asp|patchinfo.xml|version.list|
iepngfix.htc|updates.txt|patchlist.txt)
acl DENYCACHE urlpath_regex (pointblank.css|login_form.css|form.css|nouupdate.ui|ahn.ui|
3n.mh)$
acl DENYCACHE urlpath_regex (Loader|gamenotice|sources|captcha|notice|reset)
no_cache deny DENYCACHE

```



```
range_offset_limit 1 MB !DENYCACHE
uri_whitespace strip
```

¿Qué hacer con las uri (o peticiones url en nuestro caso) que contengan espacios en blanco? Strip: es decir, eliminarlos de la uri y proceder sin ellos.

```
#=====
=====
```

Reglas para bloquear contenido de anuncios en las páginas web usando expresiones regulares (las mismas que nos valdrían para *grep*, por ejemplo):

```
# Rules to block few Advertising sites
acl ads url_regex -i .youtube\.com\ad_frame?
acl ads url_regex -i .(s|s[0-90-9])\.youtube\.com
acl ads url_regex -i .googlesyndication\.com
acl ads url_regex -i .doubleclick\.net
acl ads url_regex -i ^http://googleads\.
acl ads url_regex -i ^http://(ad|ads|ads[0-90-9]|ads\d|kad|a[b|d]|ad\d|adserver|adsbox)\.[a-z0-9]*\.[a-z][a-z]*
acl ads url_regex -i ^http://openx\.[a-z0-9]*\.[a-z][a-z]*
acl ads url_regex -i ^http://[a-z0-9]*\.openx\.net/
acl ads url_regex -i ^http://[a-z0-9]*\.u-ad\.info/
acl ads url_regex -i ^http://adserver\.bs/
acl ads url_regex -i !^http://adf\.ly
http_access deny ads
http_reply_access deny ads
#deny_info http://yoursite/yourad,htm ads
#==== End Rules: Advertising ====
```

```
strip_query_terms off
```

```
acl yutub url_regex -i .*youtube\.com\.*$
acl yutub url_regex -i .*youtu\.be\.*$
logformat squid1 %{Referer}>h %ru
access_log /var/log/squid/yt.log squid1 yutub
```

Lo que viene a continuación es complejo de entender y es una característica propia de la versión 2.7 (stable) de squid. Es específico del “cacheo” de contenido en streaming cambiante. Es decir, cuando se accede a un contenido en streaming como youtube u otros (según se recogen en las reglas a continuación), se genera una url temporal que apunta al contenido en ese momento, pero que cambiará en el siguiente acceso. De lo que se trata es de ofrecer al contenido cacheado una url fija si se corresponde con el mismo contenido cacheado. Es decir, que una nueva url de descarga no sea identificada por squid como contenido nuevo que no haya sido previamente almacenado en *cache*. Para esta tarea se necesitan unas reglas que definen el formato que identifica el tipo de url susceptible de ser de dinámica y un programa externo (en este caso un script en perl) que realizará dichos cambios:

<http://wiki.squid-cache.org/Features/StoreUrlRewrite>

```
#==== Custom Option REWRITE ====
```

```

acl store_rewrite_list urlpath_regex \.(get_video\?|videodownload\?|videoplayback.*id)

acl store_rewrite_list urlpath_regex \.(mp2|mp3|mid|midi|mp[234]|wav|ram|ra|rm|au|3gp|m4r|m4a)\?
acl store_rewrite_list urlpath_regex \.(mpg|mpeg|mp4|m4v|mov|avi|asf|wmv|wma|dat|flv|swf)\?
acl store_rewrite_list urlpath_regex \.(jpeg|jpg|jpe|jp2|gif|tiff?|pcx|png|bmp|pic|ico)\?
acl store_rewrite_list urlpath_regex \.(chm|dll|doc|docx|xls|xlsx|ppt|pptx|pps|ppsx|mdb|mdbx)\?
acl store_rewrite_list urlpath_regex \.(txt|conf|cfm|psd|wmf|emf|vsd|pdf|rtf|odt)\?
acl store_rewrite_list urlpath_regex \.(class|jar|exe|gz|bz|bz2|tar|tgz|zip|gzip|arj|ace|bin|cab|msi|rar)\?
acl store_rewrite_list urlpath_regex \.(htm|html|mhtml|css|js)\?

acl store_rewrite_list_web url_regex ^http:\V\([A-Za-z-]+[0-9]+)*\.[A-Za-z]*\.[A-Za-z]*
acl store_rewrite_list_web_CDN url_regex ^http:\V\[a-z]+[0-9]\.google\.com doubleclick\.net

acl store_rewrite_list_path urlpath_regex \.(mp2|mp3|mid|midi|mp[234]|wav|ram|ra|rm|au|3gp|m4r|m4a)$
acl store_rewrite_list_path urlpath_regex \.(mpg|mpeg|mp4|m4v|mov|avi|asf|wmv|wma|dat|flv|swf)$
acl store_rewrite_list_path urlpath_regex \.(jpeg|jpg|jpe|jp2|gif|tiff?|pcx|png|bmp|pic|ico)$
acl store_rewrite_list_path urlpath_regex \.(chm|dll|doc|docx|xls|xlsx|ppt|pptx|pps|ppsx|mdb|mdbx)$
acl store_rewrite_list_path urlpath_regex \.(txt|conf|cfm|psd|wmf|emf|vsd|pdf|rtf|odt)$
acl store_rewrite_list_path urlpath_regex \.(class|jar|exe|gz|bz|bz2|tar|tgz|zip|gzip|arj|ace|bin|cab|msi|rar)$
acl store_rewrite_list_path urlpath_regex \.(htm|html|mhtml|css|js)$

acl getmethod method GET

storeurl_access deny !getmethod
#this is not related to youtube video its only for CDN pictures
storeurl_access allow store_rewrite_list_web_CDN
storeurl_access allow store_rewrite_list_web store_rewrite_list_path
storeurl_access allow store_rewrite_list
storeurl_access deny all
storeurl_rewrite_program /etc/squid/storeurl.pl
storeurl_rewrite_children 10
storeurl_rewrite_concurrency 40
# ==== End Custom Option REWRITE ====

#=====
# Custom Option REFRESH PATTERN
#=====
refresh_pattern (get_video\?|videoplayback\?|videodownload\?|\.flv\?|\.fid\?) 43200 99% 43200
override-expire ignore-reload ignore-must-revalidate ignore-private
refresh_pattern -i (get_video\?|videoplayback\?|videodownload\?) 5259487 999% 5259487

```

```

override-expire ignore-reload reload-into-ims ignore-no-cache ignore-private
# -- refresh pattern for specific sites -- #
refresh_pattern ^http://*.jobstreet.com.*.* 720 100% 10080 override-expire override-lastmod
ignore-no-cache
refresh_pattern ^http://*.indowebster.com.*.* 720 100% 10080 override-expire override-lastmod
reload-into-ims ignore-reload ignore-no-cache ignore-auth
refresh_pattern ^http://*.21cinplex.*.* 720 100% 10080 override-expire override-lastmod reload-
into-ims ignore-reload ignore-no-cache ignore-auth
refresh_pattern ^http://*.atmajaya.*.* 720 100% 10080 override-expire ignore-no-cache ignore-
auth
refresh_pattern ^http://*.kompas.*.* 720 100% 10080 override-expire override-lastmod reload-
into-ims ignore-no-cache ignore-auth
refresh_pattern ^http://*.theinquirer.*.* 720 100% 10080 override-expire ignore-no-cache ignore-
auth
refresh_pattern ^http://*.blogspot.com/. * 720 100% 10080 override-expire override-lastmod
reload-into-ims ignore-no-cache ignore-auth
refresh_pattern ^http://*.wordpress.com/. * 720 100% 10080 override-expire override-lastmod
reload-into-ims ignore-no-cache
refresh_pattern ^http://*.photobucket.com/. * 720 100% 10080 override-expire override-lastmod
reload-into-ims ignore-no-cache ignore-auth
refresh_pattern ^http://*.tinypic.com/. * 720 100% 10080 override-expire override-lastmod reload-
into-ims ignore-no-cache ignore-auth
refresh_pattern ^http://*.imageshack.us/. * 720 100% 10080 override-expire override-lastmod
reload-into-ims ignore-no-cache ignore-auth
refresh_pattern ^http://*.kaskus.*.* 720 100% 28800 override-expire override-lastmod reload-
into-ims ignore-no-cache ignore-auth
refresh_pattern ^http://www.kaskus.com/. * 720 100% 28800 override-expire override-lastmod
reload-into-ims ignore-no-cache ignore-auth
refresh_pattern ^http://*.detik.*.* 720 50% 2880 override-expire override-lastmod reload-into-ims
ignore-no-cache ignore-auth
refresh_pattern ^http://*.detiknews.*.* 720 50% 2880 override-expire override-lastmod reload-
into-ims ignore-no-cache ignore-auth
refresh_pattern ^http://video.liputan6.com/. * 720 100% 10080 override-expire override-lastmod
reload-into-ims ignore-no-cache ignore-auth
refresh_pattern ^http://static.liputan6.com/. * 720 100% 10080 override-expire override-lastmod
reload-into-ims ignore-no-cache ignore-auth
refresh_pattern ^http://*.friendster.com/. * 720 100% 10080 override-expire override-lastmod
ignore-no-cache ignore-auth
refresh_pattern ^http://*.facebook.com/. * 720 100% 10080 override-expire override-lastmod
reload-into-ims ignore-no-cache ignore-auth
refresh_pattern ^http://apps.facebook.com/. * 720 100% 10080 override-expire override-lastmod
reload-into-ims ignore-no-cache ignore-auth
refresh_pattern ^http://*.fbcdn.net/. * 720 100% 10080 override-expire override-lastmod reload-
into-ims ignore-no-cache ignore-auth
refresh_pattern ^http://profile.ak.fbcdn.net/. * 720 100% 10080 override-expire override-lastmod
reload-into-ims ignore-no-cache ignore-auth
refresh_pattern ^http://static.playspoon.com/. * 720 100% 10080 override-expire override-lastmod
reload-into-ims ignore-no-cache ignore-auth

```

```

refresh_pattern ^http://cooking.game.playspoon.com/. * 720 100% 10080 override-expire override-
lastmod reload-into-ims ignore-no-cache ignore-auth
refresh_pattern -i http://[^a-z\.]*onemanga\.com/? 720 80% 10080 override-expire override-
lastmod reload-into-ims ignore-no-cache ignore-auth
refresh_pattern ^http://media?.onemanga.com/. * 720 80% 10080 override-expire override-lastmod
reload-into-ims ignore-no-cache ignore-auth
refresh_pattern ^http://*.yahoo.com/. * 720 80% 10080 override-expire override-lastmod reload-
into-ims ignore-no-cache ignore-auth
refresh_pattern ^http://*.google.com/. * 720 80% 10080 override-expire override-lastmod reload-
into-ims ignore-no-cache ignore-auth
refresh_pattern ^http://*.forummikrotik.com/. * 720 80% 10080 override-expire override-lastmod
reload-into-ims ignore-no-cache ignore-auth
refresh_pattern ^http://*.linux.or.id/. * 720 100% 10080 override-expire override-lastmod reload-
into-ims ignore-no-cache ignore-auth
# -- refresh pattern for extension -- #
refresh_pattern -i \.(mp2|mp3|mid|midi|mp[234]|wav|ram|ra|rm|au|3gp|m4r|m4a)(\?.*|$) 5259487
999% 5259487 override-expire ignore-reload reload-into-ims ignore-no-cache ignore-private
refresh_pattern -i \.(mpg|mpeg|mp4|m4v|mov|avi|asf|wmv|wma|dat|flv|swf)(\?.*|$) 5259487 999%
5259487 override-expire ignore-reload reload-into-ims ignore-no-cache ignore-private
refresh_pattern -i \.(jpeg|jpg|jpe|jp2|gif|tiff?|pcx|png|bmp|pic|ico)(\?.*|$) 5259487 999% 5259487
override-expire ignore-reload reload-into-ims ignore-no-cache ignore-private
refresh_pattern -i \.(chm|dll|doc|docx|xls|xlsx|ppt|pptx|pps|ppsx|mdb|mdbx)(\?.*|$) 5259487 999%
5259487 override-expire ignore-reload reload-into-ims ignore-no-cache ignore-private
refresh_pattern -i \.(txt|conf|cfm|psd|wmf|emf|vsd|pdf|rtf|odt)(\?.*|$) 5259487 999% 5259487
override-expire ignore-reload reload-into-ims ignore-no-cache ignore-private
refresh_pattern -i \.(class|jar|exe|gz|bz|bz2|tar|tgz|zip|gzip|arj|ace|bin|cab|msi|rar)(\?.*|$)
5259487 999% 5259487 override-expire ignore-reload reload-into-ims ignore-no-cache ignore-
private
refresh_pattern -i \.(htm|html|mhtml|css|js)(\?.*|$) 1440 90% 86400 override-expire ignore-reload
reload-into-ims
#=====
=====
refresh_pattern -i (/cgi-bin/|\?) 0 0% 0
refresh_pattern ^gopher: 1440 0% 1440
refresh_pattern ^ftp: 10080 95% 10080 override-lastmod reload-into-ims
refresh_pattern . 0 20% 10080 override-lastmod reload-into-ims

```

A continuación se define qué hacer con las ACL definidas al principio. Son la ACL propiamente dichas:

```

http_access allow manager localhost
http_access deny manager
http_access allow purge localhost
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports

```

```

http_access allow localnet
http_access allow all
http_access deny all

```

```
icp_access allow localnet
```

```
icp_access deny all
```

*icp\_port 0* #Esto significa que el puerto no está definido para la transmisión entre proxies a través del protocolo ICP. Es decir, no se espera que haya otros proxies en estructura “padre/hijos (parent/child)” o “hermanos (siblings)”.

```
buffered_logs on
```

```
acl shoutcast rep_header X-HTTP09-First-Line ^ICY.[0-9]
```

```
upgrade_http0.9 deny shoutcast
```

```
acl apache rep_header Server ^Apache
```

```
broken_vary_encoding allow apache
```

Reglas específicas para que los servidores web no detecten al proxy y que este sea, efectivamente, transparente:

```
forwarded_for off
```

```
header_access From deny all
```

```
header_access Server deny all
```

```
header_access Link deny all
```

```
header_access Via deny all
```

```
header_access X-Forwarded-For deny all
```

```
httpd_suppress_version_string on
```

```
shutdown_lifetime 10 seconds
```

```
snmp_port 3401
```

```
snmp_access allow snmppublic all
```

```
dns_timeout 1 minutes
```

```
dns_nameservers 8.8.8.8 8.8.4.4
```

```
fqdn_cache_size 5000 #16384
```

```
ip_cache_size 5000 #16384
```

```
ip_cache_low 98
```

```
ip_cache_high 99
```

```
log_fqdn off
```

```
log_icp_queries off
```

```
memory_pools off
```

```
maximum_single_addr_tries 2
```

```
retry_on_error on
```

```
icp_hit_stale on
```

```
strip_query_terms off
```

```

query_icmp on
reload_into_ims on
emulate_httpd_log off
negative_ttl 0 seconds
pipeline_prefetch on
vary_ignore_expire on
half_closed_clients off
high_page_fault_warning 2
nonhierarchical_direct on
prefer_direct off

```

Definimos la cuenta de correo electrónico que aparecerá como la del administrador al que envía correos cuando al cliente le muestre squid algún mensaje de error:

```
cache_mgr fj.detorres@gmail.com
```

Definición de usuario y grupo que tiene acceso al proxy. Creados previamente en la instalación de squid. Es el equivalente al usuario `www-data` creado por Apache en su instalación. En conjunción con un servidor Apache instalado en la misma máquina, se puede definir a `www-data` también como usuario con privilegios en `squid` (interesante para el uso conjunto de `squid` con otras herramientas como `squidcachewiwer` que emplean tecnología web para mostrar su contenido):

```

cache_effective_user proxy
cache_effective_group proxy
visible_hostname strawberrylabs.
unique_hostname raspberry_pi

```

```
cachemgr_passwd none all
```

Contraseña para realizar diversas acciones del contenido cacheado, las cuales han de especificarse en este archivo de configuración, de entre esta lista:

```

5min
60min
asndb
authenticator
cbdata
client_list
comm_incoming
config *
counters
delay
digest_stats
dns
events
filedescriptors
fqdn_cache
histograms
http_headers
info
io
ipcache
mem

```

```

menu
netdb
non_peers
objects
offline_toggle *
pconn
peer_select
reconfigure *
redirector
refresh
server_list
shutdown *
store_digest
storedir
utilization
via_headers
vm_objects

```

Se ha establecido que no se requiera contraseña para ninguna de ellas. Es lo adecuado para un proxy que pretende ser transparente.

```

client_db on
max_filedescriptors 8192

```

Posibilidad de recoger estadísticas de conexión de los distintos clientes, habilitada.

```

# ZPH config Marking Cache Hit, so cached contents can be delivered at full lan speed via MT
zph_mode tos
zph_local 0x30
zph_parent 0
zph_option 136

```

## Configuración del sistema para que el proxy sea transparente al usuario

Vamos a utilizar iptables en conjunción con el paquete bridge-utils con el que generaremos un puente entre las dos tarjetas, y que en Moebius no viene instalado, para capturar todo el tráfico entrante por eth0 y que salga por eth1, haciendo que las conexiones pasen por Squid. Para lo cual habrá que redirigir el tráfico a los puertos de escucha del squid.

Ha de tenerse en cuenta que la configuración tanto de squid.conf como de nuestro script de enrutamiento tienen en cuenta que la ip asignada a la Raspberry Pi es la 192.168.1.4, puesto que, en las redes LAN domésticas, los routers suelen estar en la 192.168.1.0/24 y suelen asignarse Ips por DHCP partir de la 192.168.1.33.

La instalacion de los paquetes necesarios es:

```
sudo apt-get install bridge-utils iptables
```

Y la configuración elegida para el archivo /etc/network/interfaces es:

```
auto lo br0
```

```

iface lo inet loopback
pre-up iptables-restore < /etc/iptables.bridge.rules

allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf

iface eth0 inet manual
iface eth1 inet manual
allow-hotplug eth1
iface br0 inet static
bridge_ports eth0 eth1
address 192.168.1.4
network 192.168.1.0
netmask 255.255.255.0
broadcast 192.168.1.255
gateway 192.168.1.1

```

En él se tiene en cuenta la existencia de una tarjeta inalámbrica, que en principio el sistema detectaría como wlan0 y un pequeño script /etc/iptables.bridge.rules que en su momento no pareció dar ningún resultado aunque tampoco ha sido retirado. Su contenido es éste:

```
-A PREROUTING -i br0 -p tcp --dport 80 -j REDIRECT --to-port 8080 COMMIT
```

La configuración elegida en nuestro hardware es tener dos tarjetas de red: eth0, que es la que tiene la Raspberry Pi, y eth1 que se consigue con un adaptador para usb. Además, si va a tratarse de nuestro sistema de cacheo portátil, podremos prescindir del transformador de corriente y reemplazarlo por un cable de conexión USB de los que se usan típicamente para conectar un teléfono móvil al ordenador. Un cable ethernet de salida de la Raspberry Pi a nuestro puerto rt45 el ordenador y el transformador USB-ethernet. Además, por supuesto, del pendrive o disco duro externo y la tarjeta SD.

Para redirigir el tráfico a través de iptables he creado un script que luego ha de ser copiado o movido a la carpeta /etc/init.d/ con la intención de que ese script sea ejecutado al inicio y se apliquen las reglas de redireccionamiento automáticamente, además de lanzar squid. Se llama *capture\_connection.sh*

```

#!/bin/bash
#Francisco Jose de Torres Sanchez-Simon
#Borramos la tabla nat en caso de que halla información en ella
#(caso en que este script ya haya sido lanzado con anterioridad):
iptables -t nat -F
#Reglas de enrutado para que todo el tráfico pase por el proxy:
iptables -t nat -A PREROUTING -i br0 -p tcp --dport 80 -j DNAT --to 192.168.1.4:8080
#Está comentadas líneas para el puerto 443 porque no se puede cachear la conexión ssl
#(tendríamos que suplir los certificados de cada página):
#iptables -t nat -A PREROUTING -i br0 -p tcp --dport 443 -j DNAT --to 192.168.1.4:8080

```



```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 8080
#iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 443 -j REDIRECT --to-port 8080
squid
ps aux | grep "squid"
```

Para que nuestro nuevo script sea ejecutable al inicio del sistema, se hace lo siguiente como root o pi desde sudo:

```
sudo chmod +x /etc/init.d/capture_connection.sh
```

```
sudo update-rc.d capture_connection.sh defaults
```

### **Instalación de Putty en Windows:**

Algo de lo que carece la Raspberry Pi es de un botón de apagado; por lo que es necesario hacerlo desde el propio sistema operativo. Si lo que queremos es apagar la Raspberry, tendremos que conectarnos por ssh y realizar algunas de esas acciones:

```
sudo halt
ó
sudo init 0
ó
sudo shutdown -h now
```

Que son todas maneras equivalentes de apagar un sistema linux. Los sistemas linux ya vienen con algún cliente ssh instalado, pero en Windows habría que instalar alguno. El más típico es putty.

### **Borrar la memoria caché:**

Para poder borrar la caché he creado el script *delete\_cache.sh* cuyo código es este:

```
#!/bin/bash
#Francisco José de Torres Sánchez-Simón
#Esta parte comprueba si soy root (imprescindible):

if [ $USER != root ]; then
    echo "Error: DEBES ejecutar este script como root"
    echo "Exiting..."
    exit 0
fi
if [[ $(ps aux | grep squid | grep -v "grep squid") != "" ]]; then
    echo "Cierre primero squid con squid -k shutdown"
    exit 0
fi
cd /cache-1
```

```
mkdir JUNK  
mv ?? swap.state* JUNK  
rm -fr JUNK &  
squid -z
```

# Manual del usuario

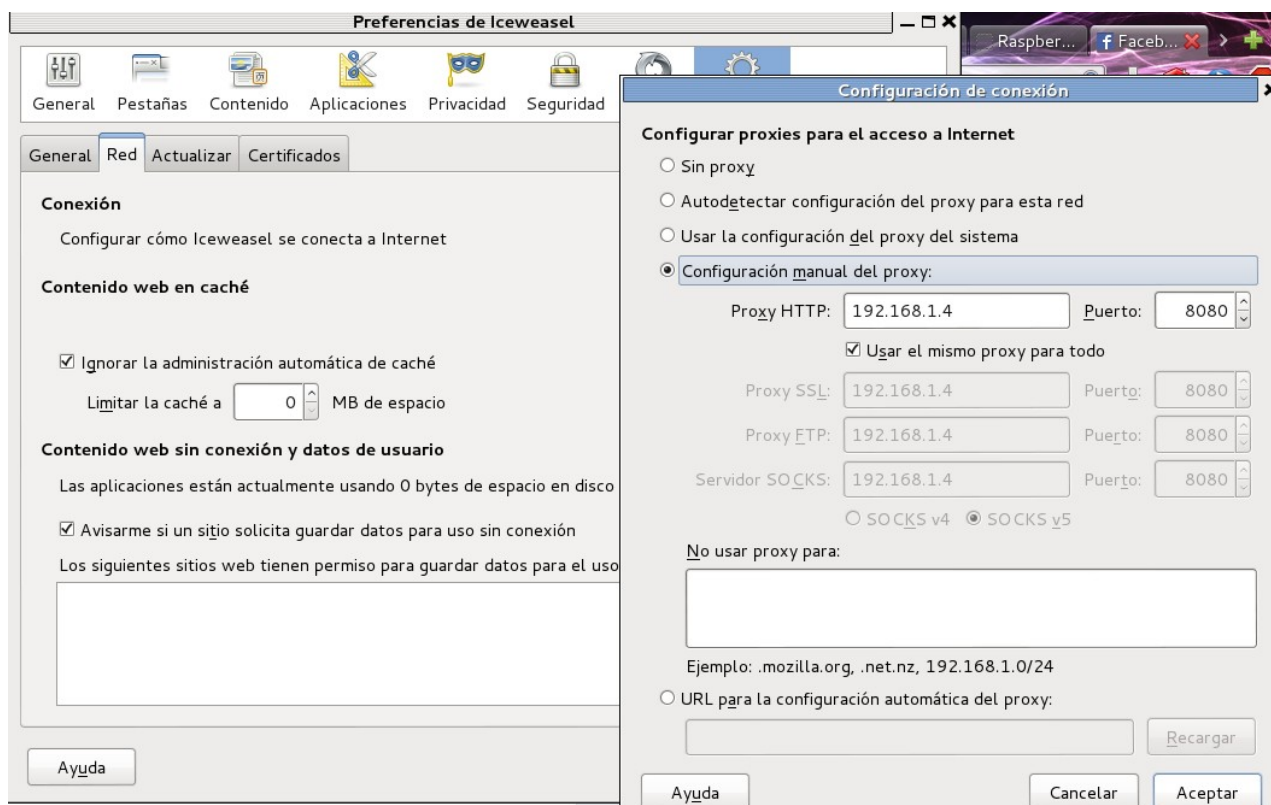
## Sin proxy transparente

Si no tenemos intención de usar el proxy como proxy transparente, tendremos que configurar los navegadores para que accedan a internet a través de él. Para anular la ejecución del script, lo más socorrido es quitarle los permisos de ejecución así:

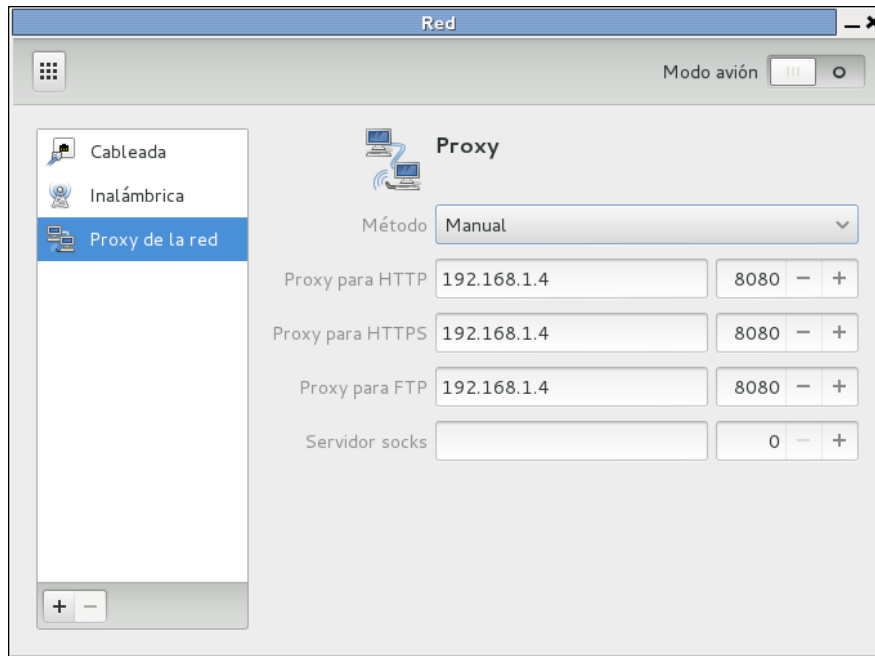
```
sudo chmod -x /etc/init.d/capture_connection.sh
```

A pesar de no ser nada elegante. Es tan sólo en el caso de que hayamos decidido que, temporalmente no se ejecute.

Al navegador firefox lo configuraríamos así:



En el caso del resto del sistema, ha de configurarse la conexión global internet, gráficamente, de esta manera:



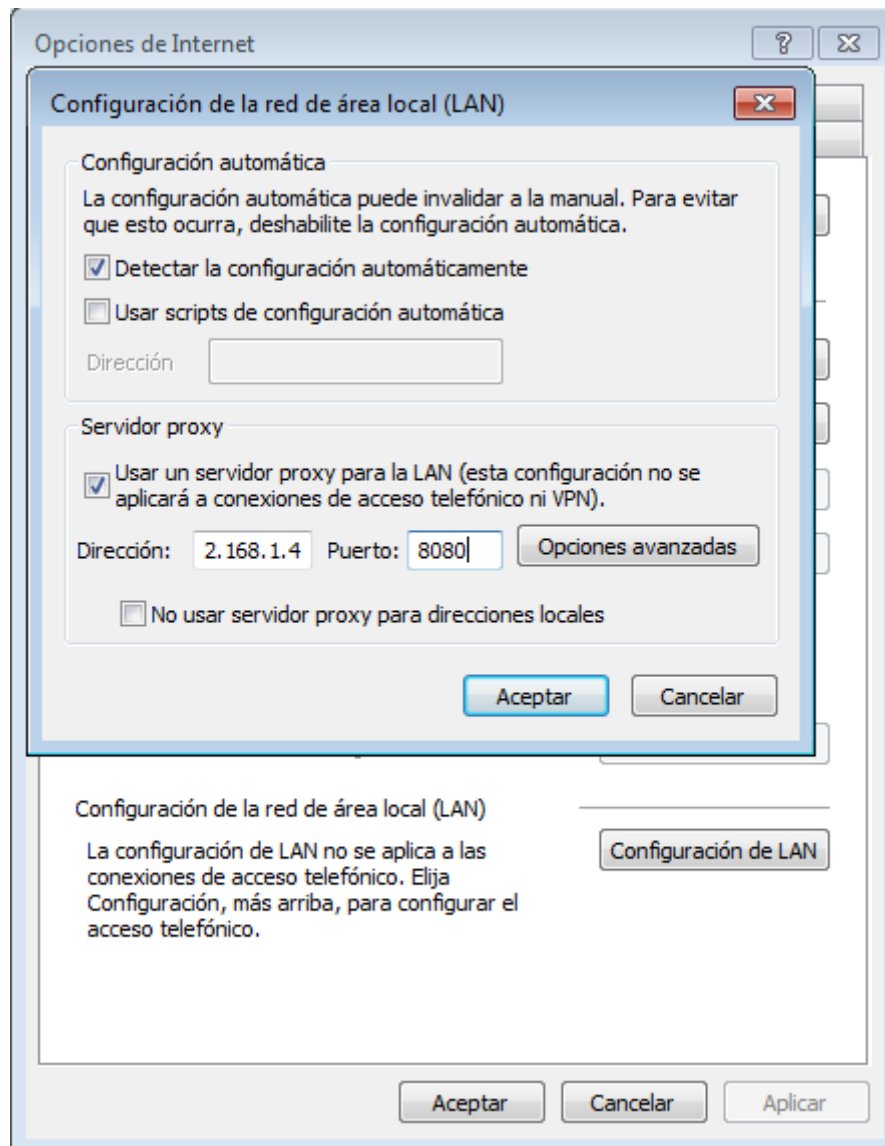
O, a través de la línea de comandos, como root:

```
# export http_proxy=http://192.168.1.4:8080
```

```
# export https_proxy=http://192.168.1.4:8080
```

```
# export ftp_proxy=http://192.168.1.4:8080
```

Para Windows, la configuración global sería:



## Con proxy transparente

Lo bueno del proxy transparente es que el usuario (o el administrador, una vez configurado todo), no tienen que hacer absolutamente nada.

**Bibliografía:**

- VNC SERVER EN RASPBERR PI:
  - <http://forum.stmlabs.com/showthread.php?tid=10384>
  - <http://raspberrypi.stackexchange.com/questions/9311/how-to-install-lxde-on-raspbmc>
- AP
  - <http://learn.adafruit.com/setting-up-a-raspberry-pi-as-a-wifi-access-point/install-software>
  - <http://www.savagehomeautomation.com/raspi-airlink101>
  - <http://www.themagpi.com/issue/issue-11/article/turn-your-raspberry-pi-into-a-wireless-access-point/>
  - <http://www.raspberrypi.org/phpBB3/viewtopic.php?f=46&t=25921>
  - <http://www.pi-point.co.uk/>
  - <http://www.raspberrypi.org/archives/2929>
  - <http://www.maketecheasier.com/set-up-raspberry-pi-as-wireless-access-point/>
  - <http://willhaley.com/blog/raspberry-pi-hotspot-ew7811un-rtl8188cus/>
- STRABERRY\_LABS :
  - <http://www.instructables.com/id/How-to-make-a-WiFi-Access-Point-out-of-a-Raspberry/>
  - Resolución del problema con la edimax:
    - <https://forums.adafruit.com/viewtopic.php?f=19&t=47716>
- RASPBIAN:
  - <https://github.com/hifi/raspbian-ua-netinst>
  - [www.raspberrypi.org](http://www.raspberrypi.org)
- RASPBIAN y su instalación rápida con BERRYBOOT:
  - <https://sites.google.com/site/raspispain/otros/instalacion-de-raspbian>
- RASPBERRY PI EMULATOR PARA WINDOWS:
  - <http://sourceforge.net/projects/rpiqemuwindows/?source=recommended>
- CONFIGURANDO RASPBERRY:
  - <http://www.ubuntumax.com/2013/01/el-home-server-perfecto-con-una.html>
- Cambiando sistema a USB:
  - <http://www.raspberrypi.org/forums/viewtopic.php?f=29&t=44177>
- INSTALANDO CLIENTE TORRENT:
  - <http://www.molesybits.es/2014/01/raspberry-owncloud-transmission.html>
- SCRIPTS:
  - <https://github.com/recantha/redwing-pi>
- Configuración de Squid:
  - Página oficial: <http://www.squid-cache.org/>
  - <http://www.comfsm.fm/computing/squid/FAQ.html#toc9>

- [http://horms.net/projects/redundant\\_linux\\_paper/related/squid/detail/acl.html](http://horms.net/projects/redundant_linux_paper/related/squid/detail/acl.html)
- SQUID:
  - <http://aacable.wordpress.com/2014/04/21/howto-cache-youtube-with-squid-lusca-and-bypass-cached-videos-from-mikrotik-queue/>
  - <http://www.enclu.com/2014/05/how-to-install-squid-on-debian-7-wheezy.html>
  - <http://wiki.squid-cache.org/ConfigExamples/DynamicContent/YouTube>
  - <http://file.hsp.net.id/Linux/Proxy/squid.conf>
  - El problema de cachear contenido dinámico de youtube:
    - <http://code.google.com/p/lusca-cache/wiki/YouTubeAprilFix>
  - Interpretando los logs de Squid:
    - <http://enavas.blogspot.com.es/2009/10/analizar-los-logs-de-accesslog-de-squid.html>
    - <http://wiki.squid-cache.org/SquidFAQ/SquidLogs#access.log>
  - Las ACLs de squid:
    - [http://horms.net/projects/redundant\\_linux\\_paper/related/squid/detail/acl.html](http://horms.net/projects/redundant_linux_paper/related/squid/detail/acl.html)
    -
- JDOWNLOADER REMOTO:
  - <http://mycmdline.esnoei.com/2011/07/27/howto-jdownloader-linux-terminal-web-interface-consola-vnc-server-x11/>
- IPTABLES:
  - <http://vensign.com/como-redireccionar-trafico-nueva-ip-iptables/>
  - <http://rooteando.com/enrutamiento-e-iptables>
  - <http://www.geekytidbits.com/transparent-content-filtering-proxy/>
  - <http://es.tldp.org/Manuales-LuCAS/GARL2/garl2/x-087-2-masq.configuration.html>
  - <http://www.maketecheasier.com/set-up-raspberry-pi-as-wireless-access-point/>
  - <http://blog.davidwolinsky.com/2011/05/dangers-of-iptables-nat-with-dhcp.html>
  - <http://albertomolina.wordpress.com/2009/01/09/nat-con-iptables/>
  - <http://www.netfilter.org/documentation/HOWTO/es/NAT-HOWTO-7.html>
  - PROXY TRANSPARENTE:
    - <http://www.cyberciti.biz/tips/linux-setup-transparent-proxy-squid-howto.html>
    -
- CONEXIONES PUENTE PARA TRASFERIR TRÁFICO:
  - <http://blog.stevebaker.org/2013/02/raspberry-pi-as-transparent-squid.htm>
- CACHEAR CONEXIONES HTTPS:
  - <http://www.thedr1ver.com/2013/04/raspberry-pi-project-packet-sniffing.html>
- TRANSMISSION CLIENT:
  - <http://www.molesybits.es/2014/01/raspberry-owncloud-transmission.html>
- Conexión 3G:
  - <http://truica-victor.com/raspberry-pi-3g-tdc/>
  - <http://ubanov.wordpress.com/2011/11/22/conectar-a-internet-por-3g-umts-desde-linea-de-comando-linux/>
  - <http://superuser.com/questions/225327/how-can-i-get-wvdial-to-run-from-etc-network-interfaces>