

Servidor Proxy-caché

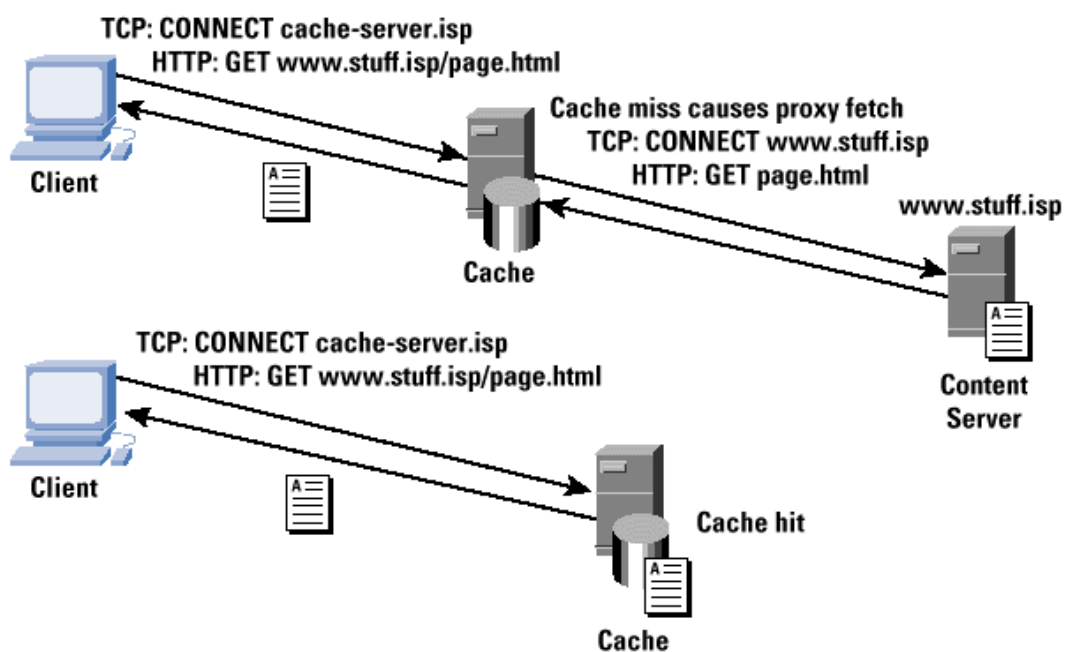
Preguntas

1. ¿En qué consiste un proxy-caché? ¿Sirve para todas las páginas? ¿En qué consiste un proxy-caché transparente?

Un servidor proxy-caché es un servidor proxy que almacena temporalmente en una caché la información solicitada, para así mejorar el tiempo de respuesta al reusar la información solicitada con frecuencia.

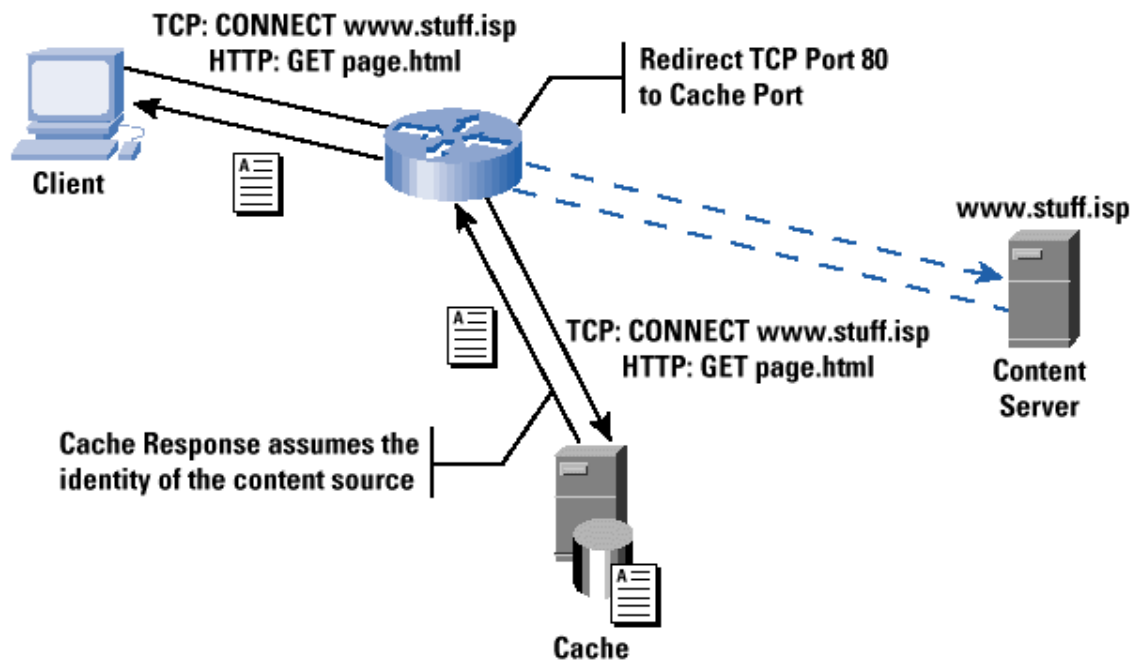
Los clientes realizan peticiones al proxy de recursos de Internet. Cuando el proxy-caché recibe la petición, busca la URL resultante en su caché local. Si la encuentra, opcionalmente contrasta la fecha y hora de la versión de la página pedida con la del servidor remoto. Si la página no ha cambiado desde que se cargo en caché la devuelve inmediatamente, ahorrándose de esta manera mucho tráfico. Si la versión es antigua o simplemente no se encuentra en la caché, lo captura del servidor remoto, lo devuelve al que lo pidió y guarda o actualiza una copia en su caché para futuras peticiones.

De su funcionamiento podemos deducir que funciona bien para contenido estático, pero no sirve para cachear páginas dinámicas, cuyo contenido genera “al vuelo” el servidor web.



Normalmente, un proxy no es transparente a la aplicación cliente: cada cliente debe ser configurado manualmente para usar el proxy. Por lo tanto, el usuario puede evadir el proxy cambiando simplemente la configuración.

Un proxy transparente combina un servidor proxy con NAT, normalmente mediante un cortafuegos, de manera que las conexiones son enrutadas dentro del proxy sin configuración por parte del cliente, y habitualmente sin que el propio cliente conozca de su existencia.



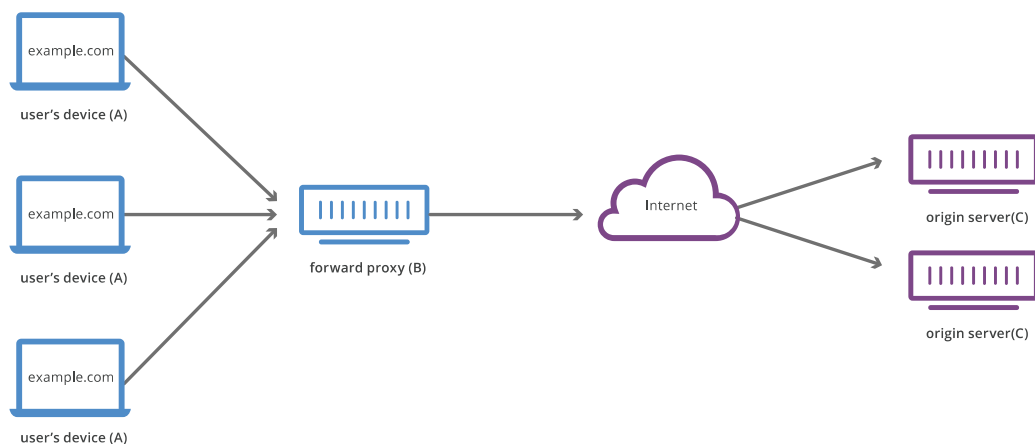
2. ¿Qué ventajas proporciona? ¿Por qué? ¿Qué desventajas proporciona? ¿Por qué?

Además de las ventajas que proporcionaba un proxy en sí, las ventajas de un proxy-caché son la descongestión del ancho de banda de la salida a internet (solicitudes repetidas de información no se sirven desde Internet) y velocidad en tiempo de respuesta (solicitudes repetidas de información se servirán a velocidad de la red local).

Además de las desventajas que proporcionaba un proxy en sí, la principal desventaja de un proxy-caché es que las páginas mostradas pueden no estar actualizadas si éstas han sido modificadas desde la última carga que realizó el proxy-caché.

3. Dibuja el esquema de una red local donde se utiliza un proxy-caché.

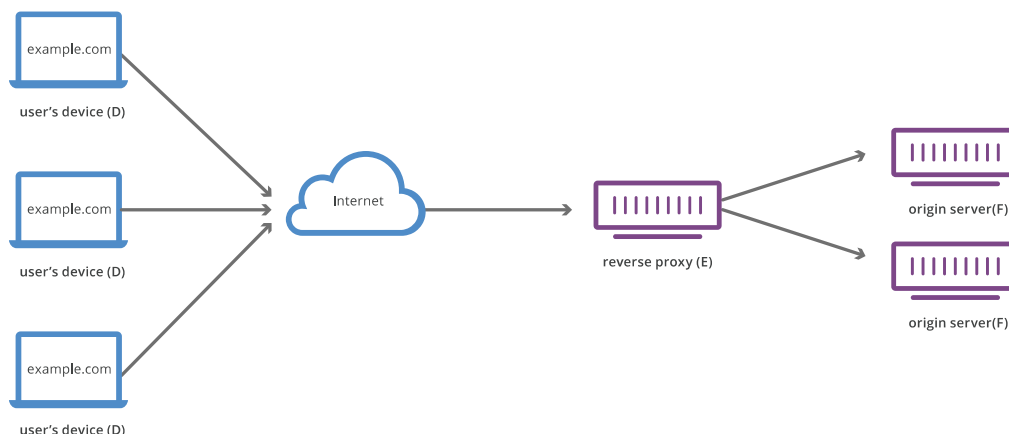
Forward Proxy Flow



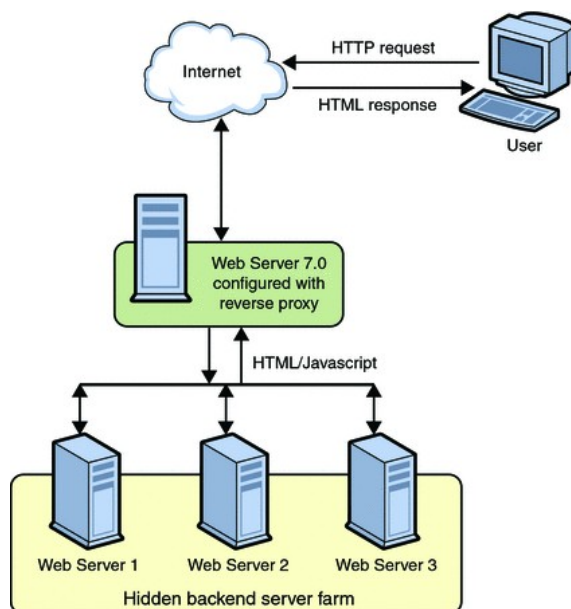
Puede estar en cualquier lugar de la red, incluso en el router. Puede ser que los clientes estén configurados para dirigir sus peticiones al proxy-caché, o que los clientes estén configurados para dirigir sus peticiones hacia internet pero que el router las redirija entonces al servidor proxy-caché de manera transparente.

4. ¿En qué consiste un proxy-caché inverso? Dibuja el esquema de una granja de servidores web donde se utiliza un proxy-caché inverso.

Reverse Proxy Flow



Un proxy inverso es un proxy o proxy-caché que no está en la red local de los clientes, sino en la red local de los servidores web. Las conexiones que vienen de internet dirigidas a los servidores web pasan a través del proxy inverso, el cual puede servir la información solicitada si estaba en su caché o pasar la petición a los servidores web que están a continuación.



Un proxy inverso nos proporciona algunas ventajas: seguridad (los servidores web quedan más aislados); aceleración de la encriptación (el proxy puede tener hardware especializado para crear las páginas seguras con SSL); balanceo de carga (el proxy puede distribuir las peticiones entre los diversos servidores web); cacheo (el proxy puede servir el contenido estático: imágenes y páginas html generadas a partir de páginas dinámicas); compresión (el proxy puede dedicarse a comprimir el contenido a servir para minimizar la cantidad de información a enviar).

5. ¿Viene algún software de proxy-caché con Windows Server? ¿Cuál es el software de proxy-caché más conocido para Windows?

No viene ningún software de proxy con Windows Server. Microsoft proporciona una solución de pago, *Microsoft ISA Server*, que entre otras cosas trae un proxy-caché para http:

<http://www.microsoft.com/forefront/edgesecurity/isaserver/>

Una lista con algunos proxy-caché para Windows la encontraréis en <http://www.web-caching.com/proxy-caches.html>. Existen alternativas gratuitas, como por ejemplo instalar *Squid* y la interficie gráfica *Kraken Config* para administrarlo.

6. ¿Cuál es el software de proxy-caché más conocido para Unix/Linux?

La solución más potente y famosa es *Squid*, pero también están bien *Varnish* y *Polipo*. Para visualizar los ficheros de logs se suele utilizar *Calamaris*.

Como proxy-caché inverso también tenemos a *Squid*, pero suele ser más sencillo configurar un servidor web como *Apache*, *Nginx* o *Lighttpd*, para que funcionen como proxy-caché inversos.

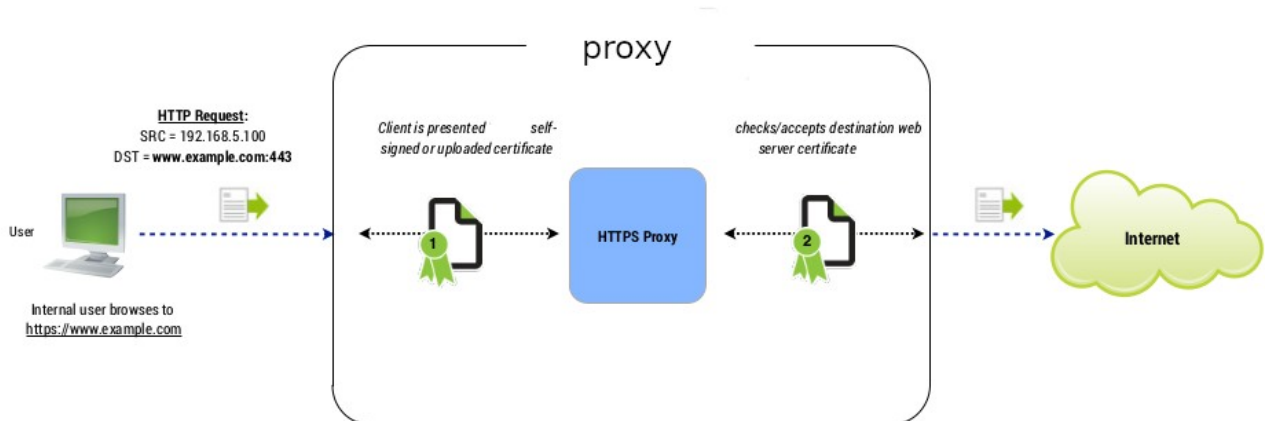
7. Al instalar un servicio de proxy-cahé, ¿Qué parámetros a configurar piensas que serán los más importantes?

Además de algunos de los parámetros que configurábamos en un proxy, tendremos:

- tamaño de la caché
- tamaño máximo del fichero a cachear
- tiempo caducidad de la información cacheada
- el algoritmo de desalojo de información de la caché cuando ésta está llena
- páginas web a filtrar

8. ¿Crees que un proxy-caché funciona de la misma manera en las peticiones HTTP que en las HTTPS?

En una petición HTTPS intervienen certificados y cifrado.



9. En la red local de nuestro centro la mayor parte del ancho de banda se la comen las actualizaciones de Windows, de antivirus, de Linux. Estudia diferentes métodos que podemos utilizar para cachear dichas actualizaciones y así liberar el ancho de banda.

- **Microsoft Windows Server Update Services:** se trata de instalar un servidor Windows que proporcionará las actualizaciones de Windows a la red local. Es un proceso complicado que requiere Active Directory.

- **IPFire , Squid** : se puede configurar Squid para que cachee las actualizaciones de MacOS y Windows, incluidas actualizaciones de antivirus y otro software (Adobe, ...)
- **Mirror o caché de paquetes Debian**: se trata de instalar un sencillo servicio que proporcionará las actualizaciones de Debian y Ubuntu a la red local. En el caso de un mirror (*apt-mirror*), se creará una imagen con todo el software disponible en los repositorios de Debian y Ubuntu. En el caso de una caché (*approx* o *apt-cacher-ng*) sólo guardará en caché del servidor las actualizaciones que se han pedido.
- **Proxy-cache para navegación y para antivirus**: se trata de instalar un proxy-caché para que el contenido web más visitado se sirvan desde la red local. Normalmente también puede cachear las actualizaciones de los antivirus, y en los antivirus se puede especificar que se actualicen desde un proxy.

Datos de la práctica

Los parámetros que configuraremos en el servidor proxy-caché son:

- 1Gb de tamaño de caché.
- Permitiremos el acceso a algunos ordenadores de nuestra red y denegaremos el resto.
- Filtraremos las páginas del sitio web <http://www.facebook.com/> y los ficheros mp3.
- Tiempo de caducidad del contenido: 1 mes.

Práctica con Windows

Existen algunas soluciones sencillas, gratuitas y poco flexibles de proxy-caché bajo Windows:

- *AnalogX Proxy* en <http://www.analogx.com/contents/download/Network/proxy/Freeware.htm>
(gratuito, pero es demasiado sencillo y no tiene caché)
- *Freeproxy* en <http://www.handcraftedsoftware.org/>
(gratuito, pero suficientemente sencillo y completo)
- *CCProxy* en <http://www.youngzsoft.net/ccproxy/>
(tiene versión de prueba limitada a tres clientes, y es complejo de configurar)
- *Wingate* en <http://www.wingate.com/>
(tiene versión de prueba limitada a un mes, y es muy completo y complejo de configurar)

Pero nosotros instalaremos el servidor de proxy-caché *Squid* para Windows Server, y también la interfaz gráfica *Kraken Config*. Exploraremos la interfaz gráfica de administración de dicho servidor, configurando los parámetros básicos. Probaremos el servidor.

1. Descargad el software y a continuación instaladlo:

<https://wiki.squid-cache.org/SquidFaq/BinaryPackages#Windows>

http://download.cnet.com/Kraken-Config-for-Squid/3000-2155_4-10495142.html

Si queréis podéis seguir los pasos que encontraréis en el siguiente tutorial, pero no hace falta porque Kraken Config ya se encarga de todo:

https://www.youtube.com/results?search_query=squid+windows

2. Configuramos el servidor proxy-caché (en una pantalla de configuración que encontraréis es importante especificar bien la dirección de la red local, en nuestro caso 192.168.100.0).
3. Probad el servicio desde un cliente. Para ello configurad el navegador para que navegue a través de la IP del proxy y del puerto 3128.

Práctica con Linux. Squid y peticiones HTTP

Aviso: para simplificar, montaremos un proxy-caché sólo para tráfico HTTP, no para HTTPS.

1. Instalamos y configuramos el servidor proxy-caché (podéis encontrar configuraciones de ejemplo muy útiles en <http://wiki.squid-cache.org/ConfigExamples/> , así como el manual de las directivas de configuración en <http://www.squid-cache.org/Doc/config/>) . Podemos bien editar el fichero /etc/squid/squid.conf, o editar un fichero en la carpeta /etc/squid/conf.d/ :

```
# apt install squid
```

```
# mv /etc/squid/squid.conf /etc/squid/squid.conf.bak
```

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

```
# Puerto para squid, caché en memoria, caché en disco
# e identificador que aparecerá en las páginas web de bloqueo
http_port 3128
cache_mem 1024 MB
cache_dir ufs /var/spool/squid 4096 16 256
visible_hostname asterix.mired.org

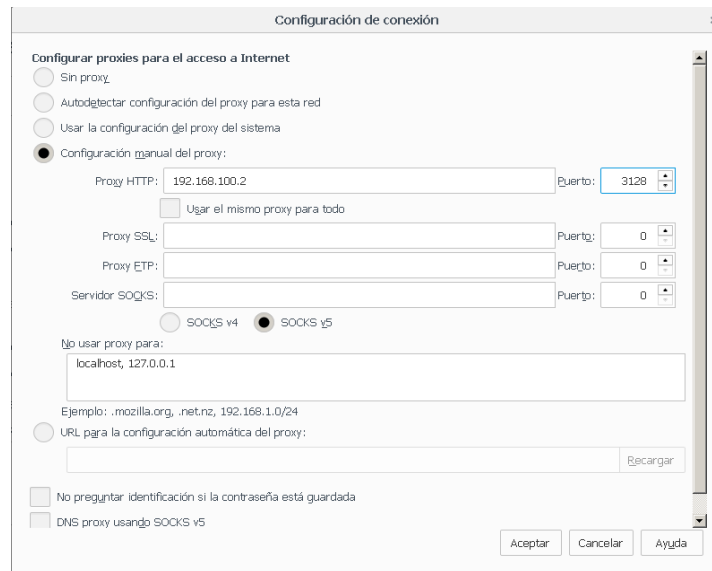
# Listas de acceso todos, localhost y mired
# En esta versión, all y localhost ya están creadas por defecto
# acl all src 0.0.0.0/0
# acl localhost src 127.0.0.1/32
acl mired src 192.168.100.0/24

# Permiso acceso a mired y localhost y deniego acceso a todos
http_access allow mired
http_access allow localhost
http_access deny all
```

y a continuación reiniciad el servicio:

```
# systemctl restart squid
```

2. Prueba el servicio desde un cliente. Para ello configura el navegador para que navegue a través de la IP del proxy y del puerto 3128.



Si el servicio parece que no funciona, prueba si ha arrancado con:

```
# ss -tlna
# tail /var/log/syslog
# systemctl status squid
# /usr/sbin/squid -k parse
```

Si el servicio ha arrancado y queremos consultar sin interfaz gráfica los logs de acceso:

```
# tail /var/log/squid/cache.log
# tail -f /var/log/squid/access.log
```

¡AVISO IMPORTANTE! Las reglas `http_access` se intentan aplicar por orden de escritura, una a una, hasta encontrar a la primera que encaja con la lista de control de acceso del ordenador que intenta acceder. Si hubiéramos escrito en primera posición `http_access deny all` ningún ordenador hubiera podido acceder, ya que todos encajan en esta lista de control de acceso. Será fundamental en los siguientes apartados que pienses bien en qué posición colocarás la regla, o no funcionará como deseas.

3. Si queremos denegar el acceso a los ordenadores de mi red, y tan solo permitir el acceso a algunos pocos, añadiremos algunas líneas más en el fichero de configuración:

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

```
# añadimos nueva lista de acceso con IPs en el fichero "permitidos"
acl permitir src "/etc/squid/permitidos"

# permitimos acceso a la nueva lista
http_access allow permitir

# borramos la línea que permitía acceso a los ordenadores de mi red
http_access allow mi red
```

y rellenaremos un fichero con las IPs de los ordenadores que dejamos acceder a internet:

```
# nano /etc/squid/permitidos
```

192.168.100.150

y a continuación reinicia el servicio y pruébalo con varios clientes con diferente IP:

systemctl restart squid

Ejemplo de cliente con navegación bloqueada por el por el proxy, pero con acceso a internet:

ERROR

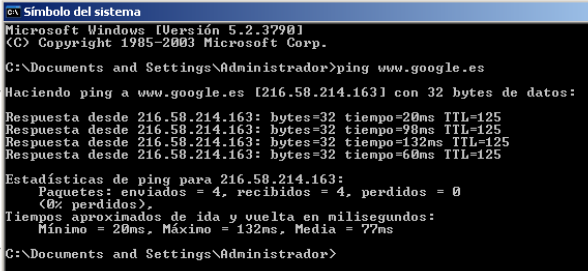
El URL solicitado no se ha podido conseguir

Se encontró el siguiente error al intentar recuperar la dirección URL: <http://www.google.es/>

Acceso Denegado

La configuración de control de acceso evita que su solicitud sea permitida en este momento. Por favor, consulte con su administrador del caché es [webmaster](#).

Generado Mon, 28 Jan 2019 13:12:56 GMT por asterix.mired.org (squid/3.5.23)



4. De forma parecida puedes bloquear el acceso a determinadas URLs, o/y a descargas de ficheros con una determinada extensión:

nano /etc/squid/squid.conf

```
# añadimos una lista de acceso con dominios y otra con expresiones
acl sitios_prohibidos dstdomain "/etc/squid/sitios_bloqueados"
acl extensiones_prohibidas url_regex -i mp3$ avi$ torrent$

# prohibimos acceso a las nuevas listas de acceso
http_access deny sitios_prohibidos
http_access deny extensiones_prohibidas
```

y rellena un fichero con las URLs de los sitios a los que no dejas acceder:

nano /etc/squid/sitios_bloqueados

```
.facebook.com
.XXX
```

y a continuación reinicia el servicio y pruébalo con varios clientes:

systemctl restart squid

5. Si queremos que el proxy-caché reenvíe las peticiones a un segundo proxy-caché, deberemos utilizar la directiva `cache_peer` (ver <http://wiki.squid-cache.org/Features/CacheHierarchy> y http://www.squid-cache.org/Doc/config/cache_peer/). En nuestro caso, suponiendo que el siguiente proxy-caché esté en la dirección IP 192.168.218.106, y que escuche en el puerto 3128, y que no utilice los protocolos HTCP e ICP de comunicación entre proxy-cachés, añadiremos al fichero de configuración de *Squid*:

nano /etc/squid/squid.conf

```
cache_peer 192.168.218.106 parent 3128 0
```

y a continuación reiniciamos el servicio:

systemctl restart squid

6. Si queremos que el proxy-caché no cachee todas la peticiones, sino que algunas de ellas no se guarden en la caché (bien porque son páginas de un servidor interno a la red y por velocidad no hace falta, o bien porque son webs dinámicas que dan algún problema al cachearlas) podemos

crear un fichero con los dominios a no cachear y añadir al fichero de configuración de *Squid*:

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

```
acl sincachear dstdomain "/etc/squid/dominios_no_cacheables"  
no_cache deny sincachear  
always_direct allow sincachear
```

y rellenaremos dicho fichero con los nombres de los dominios que el proxy no guardará en la caché (importante: sin olvidar los puntos iniciales) :

```
# nano /etc/squid/dominios_no_cacheables
```

```
.mired.org  
.gmail.com
```

y a continuación reiniciamos el servicio:

```
# systemctl restart squid
```

Con la experiencia acumulada de puntos anteriores, se deduce que también puedes crear listas de control de acceso para no cachear especificando con `url_regex` las URL de los sitios, bien sea en fichero o en expresión regular. Por ejemplo: `acl sincachear url_regex -i php$`

7. Cuando el proxy-caché recibe una petición de una página web, necesita resolver la dirección del campo *host* de la petición HTTP a IP. Para ello por defecto utiliza como servidores de DNS los valores encontrados en el fichero `/etc/resolv.conf`. Si quisiéramos utilizar otros valores para los servidores de DNS de *Squid*, deberíamos configurar la directiva `dns_nameservers` en el fichero de configuración. Por ejemplo:

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

```
dns_nameservers 127.0.0.1 192.168.100.2
```

y a continuación reiniciamos el servicio:

```
# systemctl restart squid
```

8. Si queremos que el proxy-caché sea transparente cambiaremos una línea a la configuración:

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

```
http_port 3128 transparent
```

y a continuación reinicia el servicio, y ya no hará falta configurar los clientes para que naveguen a través del proxy:

```
# systemctl restart squid
```

Para que pueda hacer de proxy-caché transparente se supone que la máquina del proxy-caché ya estaba configurada para hacer de router o puerta de enlace (ver ejercicio anterior sobre enrutamiento), y ahora sólo le indicaremos que además el tráfico dirigido al puerto 80 lo reenvíe al puerto 3128 por donde escucha por defecto el proxy-caché, añadiendo al script de enrutamiento (ver ejercicio anterior sobre enrutamiento) que se ejecutaba al inicio (en este script supongo que la tarjeta de red conectada a la red local es *eth0*):

```
sudo nano /etc/init.d/enrutamiento.sh
```

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 3128
```

9. Si queremos visualizar el estado y los logs de manera gráfica y agradable, tenemos *squidview* y

calamaris, respectivamente:

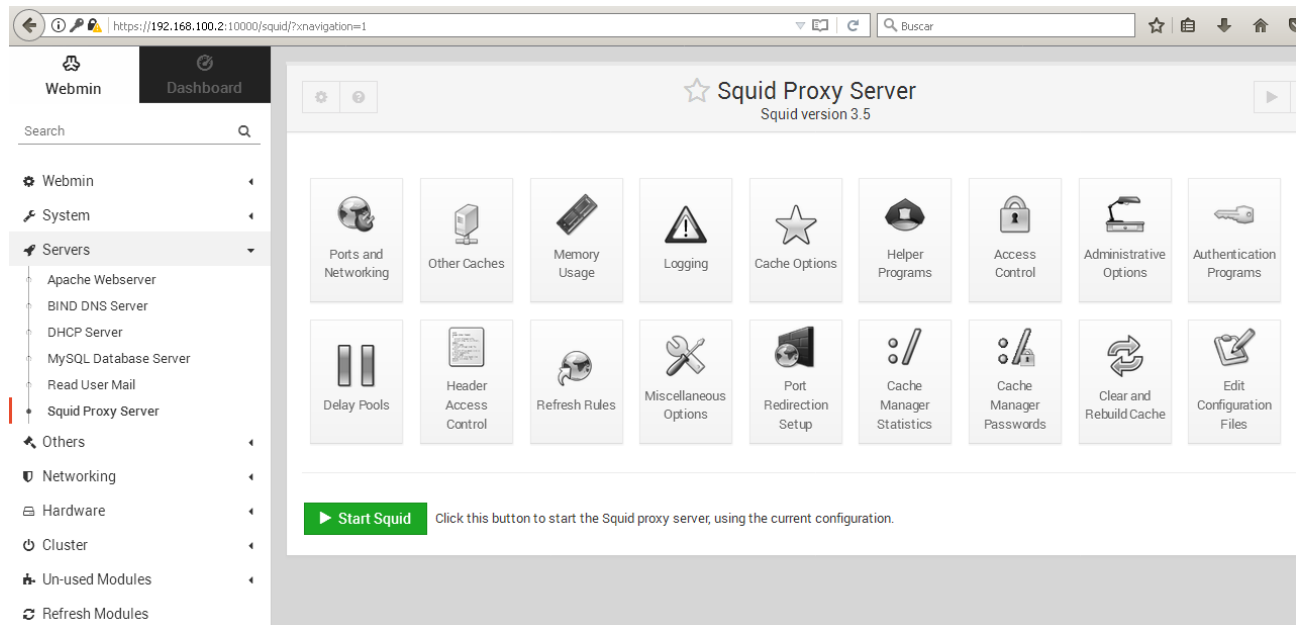
```
# apt install squidview calamaris
```

10. Y si queremos configurar el proxy-caché desde un entorno gráfico instalaremos el módulo de *webmin* correspondiente:

https://IP_servidor:10000/ → Un-used modules → Squid Proxy Server

https://IP_servidor:10000/ → Refresh modules

https://IP_servidor:10000/ → Servers → Squid Proxy Server



Práctica con Linux. Squid y peticiones HTTP

Configuraremos Squid para que actúe correctamente de proxy-caché en sitios HTTPS.

<https://www.howtoforge.com/filtering-https-traffic-with-squid>

Práctica con IPFire

En la máquina virtual que hace de router con IPFire, activa el módulo de proxy-caché y configúralo:

- <https://192.168.100.1:444/> → usuari: admin → red → web proxy → activa →

https://wiki.ipfire.org/configuration/network/proxy/wui_conf

Práctica evitar el filtrado de un proxy

Hemos protegido una red de una organización con un proxy para evitar el acceso web a páginas de piratería y hacking. Dicha red también filtra Tor y el acceso a proxys públicos. Pero una persona de dicha organización un día necesita saltarse nuestro proxy para acceder a unas páginas filtradas de seguridad informática, aprovechando que dispone de servidores en Internet (por ejemplo, Oracle Cloud y Amazon AWS).

Aunque dicha persona ya puede acceder algunas de las páginas filtradas a través de la caché de Google y el traductor de Google, con un ordenador en el exterior con IP pública una persona puede intentar evitar el proxy de la organización pasando a través de un proxy propio, una VPN propia o un tunel SSH.

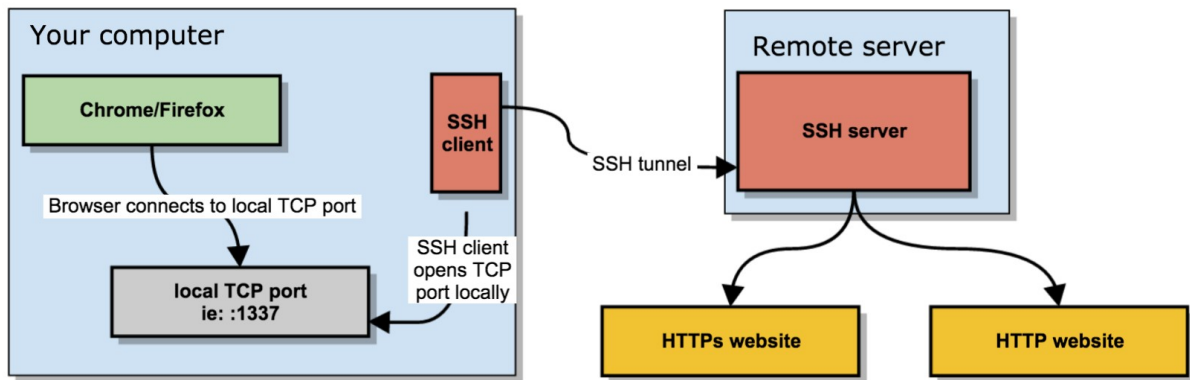
En mi caso no probé de instalar un proxy Squid escuchando en el puerto 443 de mi instancia en Oracle Cloud, ni probé la solución de crear una VPN contra la instancia en Oracle Cloud.

La solución más sencilla que encontré es aprovechar que dicho instancia en la nube tiene habilitado el acceso por SSH para crear un túnel SSH que haga de proxy SOCKS, donde las peticiones viajarán encriptadas. No hace falta instalar ningún software en la instancia:

```
$ ssh -D puerto_aleatorio -q -C -N usuario@máquina_en_Internet
```

Si no se tiene permisos de administrador en el equipo de trabajo de la organización que filtra, el puerto escogido debe estar por encima de 1024. Además en Oracle Cloud debemos especificar el fichero de la clave privada para acceder por ssh a la instancia. Por ejemplo:

```
$ ssh -D 1337 -q -C -N ubuntu@IP_Oracle_Cloud -i fichero_clave
```



A continuación hay que configurar nuestro navegador que utilice un proxy SOCKS en la IP de localhost y el puerto escogido. El tráfico se dirigirá por el tunel SSH.

Configure Proxies to Access the Internet

☐ No proxy

☐ Auto-detect proxy settings for this network

☐ Use system proxy settings

☒ Manual proxy configuration:

HTTP Proxy: Port:

☐ Use this proxy server for all protocols

SSL Proxy: Port:

FTP Proxy: Port:

SOCKS Host: Port:

☐ SOCKS v4 ☒ SOCKS v5

No Proxy for:

Example: .mozilla.org, .net.nz, 192.168.1.0/24

☐ Automatic proxy configuration URL:

☐ Do not prompt for authentication if password is saved

☒ Proxy DNS when using SOCKS v5

Práctica con caché de paquetes Debian

Instala y configura en uno de tus ordenadores una caché de paquetes con *apt-cacher-ng*, y configurad la red para que el resto de ordenadores con Debian o Ubuntu se actualicen a través de él:

- <https://blog.ichasco.com/apt-cache-ng-repositorio-local-cacheado/>

Referencias

- http://en.wikipedia.org/wiki/Proxy_server
- http://en.wikipedia.org/wiki/Web_cache
- http://en.wikipedia.org/wiki/Reverse_proxy
- <http://www.nexolinux.com/proxy-squid-control-de-accesos-acl-ii-2/>