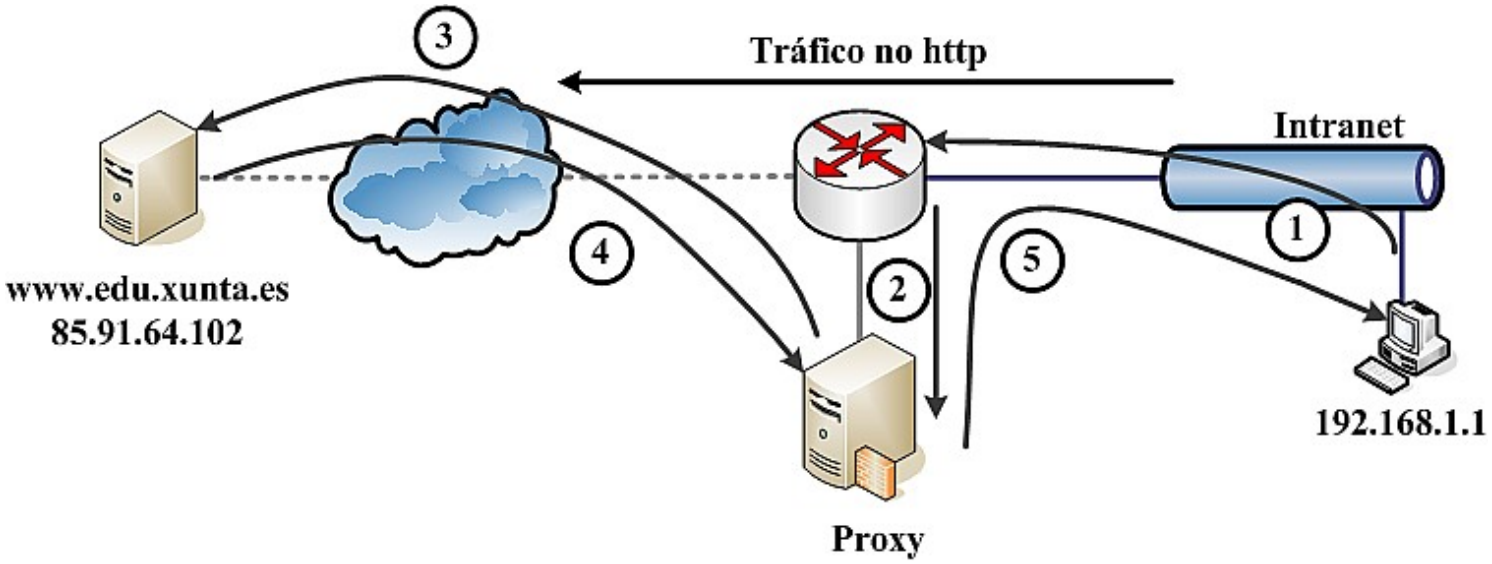
5. Proxy Transparente (Transparent Caching, Cache Redirection, Interception Caching)

Un equipo intercepta las peticiones realizadas por los clientes y las redirige al proxy. Una vez en el proxy, es éste el que se encarga de solicitar la información al servidor correspondiente. La ventaja sobre el proxy estándar es que los clientes no necesitan configurar su salida por el proxy; es decir, los clientes no son conscientes de la existencia del proxy, de ahí el nombre de transparente.



1. El cliente envía una Request al servidor web www.edu.xunta.es. Como el servidor web al que se le hace la solicitud se encuentra en otra red, el cliente utilizará su default gateway para enrutar el paquete.
2. La solicitud http es desviada silenciosamente hacia el servidor Proxy.
3. El Proxy analiza la solicitud del cliente y lanza una solicitud al servidor web www.edu.xunta.es.
4. El servidor web responde a la solicitud enviada por el Proxy.
5. El proxy prepara un paquete conteniendo la respuesta a la petición del cliente colocando como IP origen la IP del servidor web y como IP Destino la del cliente; de esta forma, el paquete al llegar al cliente parece proceder directamente del servidor web.

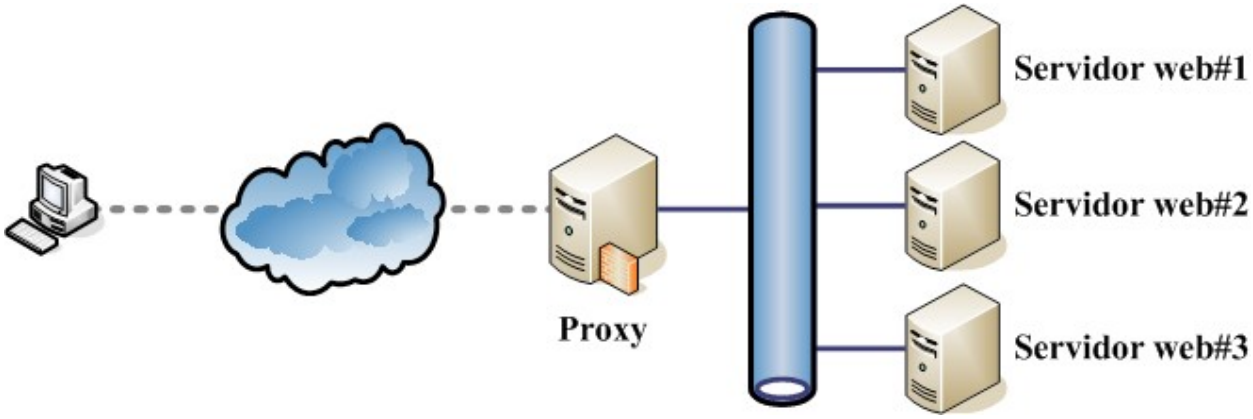
En la siguiente captura se puede ver el proceso:

```
 6 192.168.1.1      85.91.64.102    HTTP    GET /web/sites/default/themes/ceou/images/menu_background.png HTTP/1.1
11 192.168.0.254    85.91.64.102    HTTP    GET /web/sites/default/themes/ceou/images/menu_background.png HTTP/1.0
21 85.91.64.102     192.168.0.254   HTTP    HTTP/1.1 200 OK  (PNG)
31 85.91.64.102     192.168.1.1     HTTP    HTTP/1.0 200 OK  (PNG)
```

- Paquete nº 6: el cliente (192.168.1.1) lanza una request http al servidor web 85.91.64.102.
- Paquete nº 11: la solicitud ha sido reenviada silenciosamente al proxy y éste lanza una request al servidor 85.91.64.102 con IP origen la suya (192.168.0.254).
- Paquete nº 21: El servidor 85.91.64.102 responde a la solicitud del proxy (192.168.0.254).
- Paquete nº 31: El proxy construye una respuesta con IP origen la del servidor 85.91.64.102 y destino la IP del cliente 192.168.1.1 y la envía al cliente.

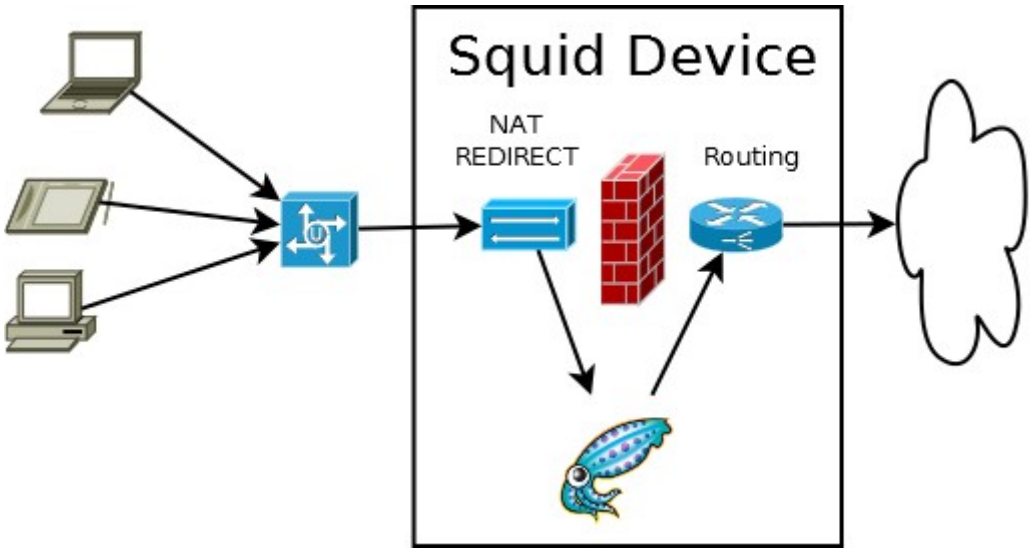
El cliente (192.168.1.1) es ajeno a la redirección http y a la solicitud realizada por el proxy. Él envía la solicitud al servidor web y recibe la respuesta del mismo. En la red de la figura, el tráfico no http sigue su camino normal sin redirección.

Aunque el no tener que configurar los clientes es una ventaja, los proxys transparentes tienen desventajas y limitaciones:



- Violación de los estándares TCP/IP: se reenvían los paquetes a un equipo que no es el destinatario, el proxy acepta paquetes no van dirigidos a él, ...
- No funciona la autenticación: los navegadores al no estar configurados para usar un proxy, rechazan enviar las credenciales a un intermediario desconocido.
- Únicamente intercepta http, dejando pasar el tráfico https.
- Los clientes web tienen que poder hacer resolución DNS.

La siguiente imagen resume de forma gráfica como funciona squid de forma transparente en un equipo Linux (en el ejemplo también es el gateway de salida a Internet):



Squid como proxy transparente. http://wiki.squid-cache.org (CC BY-SA 2.5)

Por lo tanto, para configurar squid como proxy transparente en un sistema Linux hay que hacer dos cosas: configurar squid y redirigir el tráfico.

Para indicar que squid trabaje en modo transparente hay que añadir el término 'intercept' al puerto de escucha:

```
http_port 3128
http_port 8080 intercept
```

Con esta configuración, squid trabajaría como proxy estándar atendiendo peticiones en el puerto tcp/3128 y para aquellos clientes que salen directamente (no configurados para usar un proxy), recibirá los paquetes redirigidos en el puerto tcp/8080.

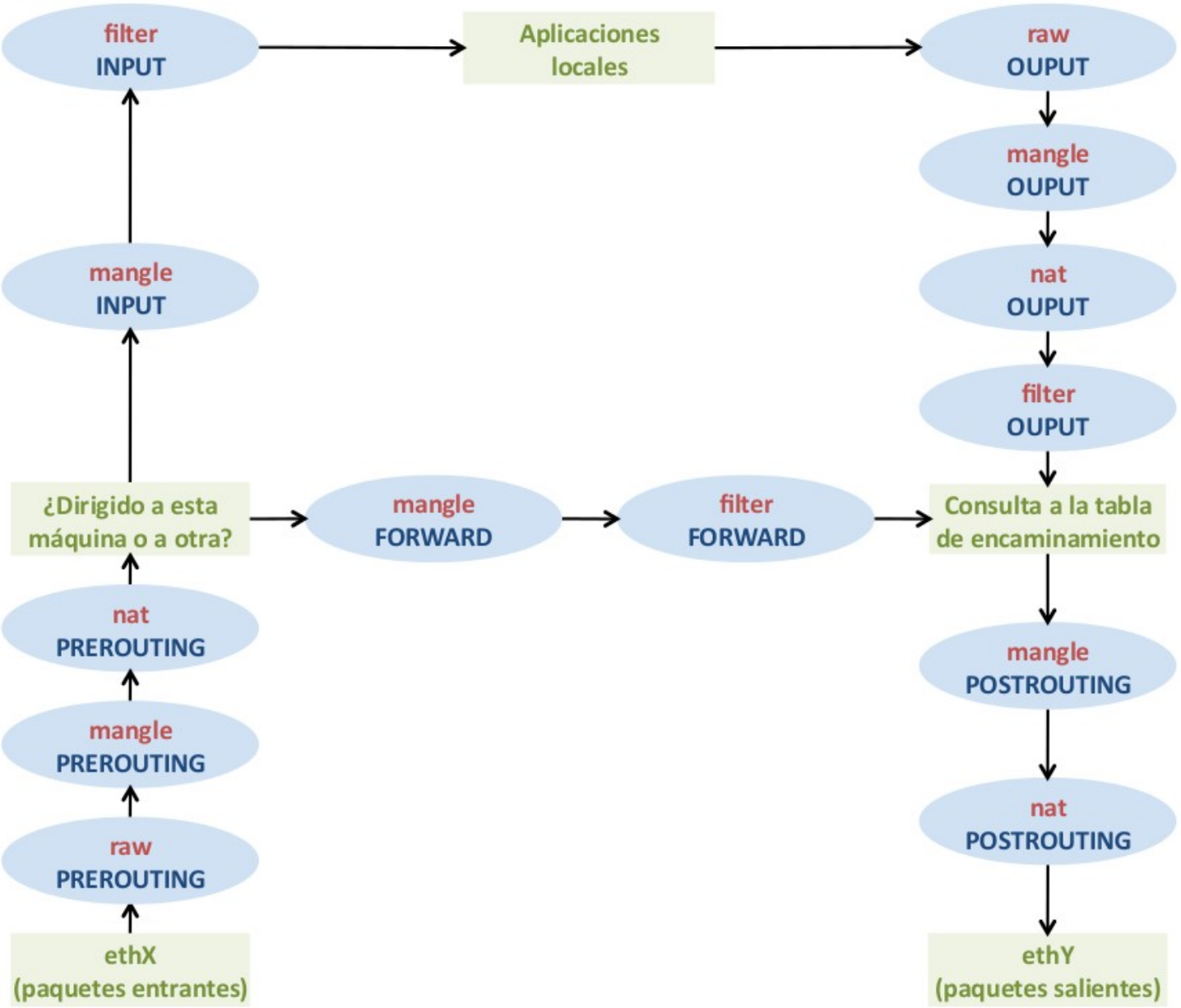
Para versiones de squid 3.3.x es obligatorio especificar dos puertos, aunque no interese trabajar en modo estándar. En caso de no indicar dos puertos habrá problemas de funcionamiento y

aparecerá el error "`No forward-proxy ports configured`" en el archivo `/var/log/squid/cache.log`. En versiones anteriores esto no era necesario.

En vez de `intercept` se puede usar `transparent`; sin embargo, los creadores de squid prefieren la nueva palabra `intercept`, ya que es más apropiada para describir este modo de funcionamiento (Interception Caching).

Para configurar netfilter para redirigir las request de los clientes hacia el servidor squid hay que tener en cuenta que en el ejemplo de la figura, las comunicaciones web de los clientes atravesarían el gateway para salir a Internet; sin embargo, esas solicitudes deben ser desviadas hacia el proceso squid corriendo en el puerto tcp/8080.

Atendiendo al esquema de procesamiento de paquetes de netfilter, se hará DNAT en PREROUTING para cambiar el destino del paquete y redirigirlo hacia el propio equipo-puerto tcp/8080.



Suponiendo que eth1 es la interfaz LAN del equipo, la redirección puede efectuarse con el comando:

```
$ sudo iptables -t nat -A PREROUTING –i eth1 -s IP_clientes -p tcp --dport 80 -j REDIRECT
--to-port 8080
```

`-j REDIRECT` es un caso particular de DNAT, donde el destino es el propio equipo. Sería equivalente a hacerlo con `-j DNAT`:

```
$ sudo iptables -t nat -A PREROUTING –i eth1 -s IP_clientes -p tcp --dport 80 -j DNAT –to IP_squid:8080
```

El paquete modificado seguirá procesándose por la cadena INPUT y habrá que autorizarlo para que llegue al proceso squid:

```
$ sudo iptables -A INPUT –i eth1 -s IP_clientes -p tcp --dport 8080 –j ACCEPT
```