Making Web Access Internet Protocol Version Agnostic

Introduction

Currently IPv6 traffic constitutes less than 1% of overall web traffic, but over the next few years we can expect an increase in the amount of web content made available via IPv6. Indeed, World IPv6 day in June 2011 saw a relatively large upturn in IPv6 traffic and following the impending IPv6 Launch Day in June 2012[i], the number of IPv6 enabled sites is expected to rise dramatically.

It is likely that initially the majority of all web content will be made available using dual-stacked servers which will directly support both IPv4 and IPv6 clients. However this approach is neither desirable nor sustainable in the longer term given the increased infrastructure complexity and the burgeoning numbers of nodes that need access in our "internet of things". At some point in the future all new websites will be accessible only via IPv6.

Equally there are many legacy devices for which HTTP access will only ever be supported using IPv4.

So what options do you have to ensure continued HTTP access to the entire world-wide web irrespective of the clients you use and the websites you wish to visit?

Fortunately help is at hand from the Squid proxy project. Squid is configurable to support many different proxying roles[ii], but in one of its configurations it can be used as an effective HTTP proxy providing content caching, access control facilities and internet protocol version translation.

Pre-requisites

So before we consider Squid in detail, it is important to discuss the system you are likely to deploy it on in this role. A typical system is outlined in Figure 1. The gateway server is required to have internet access via both IPv4 and IPv6, and along with the obvious HTTP proxying role it may additionally provide multiple services to its clients including:

1. An IPv4/IPv6 DNS facility
2. DHCP, v4 and potentially v6 too, dependent upon your preferred address allocation method
3. Router advertisements supporting SLAAC, if IPv6 addresses are not allocated via DHCPv6

These services could also be provided via different machines/nodes but they must exist somewhere in your client network.

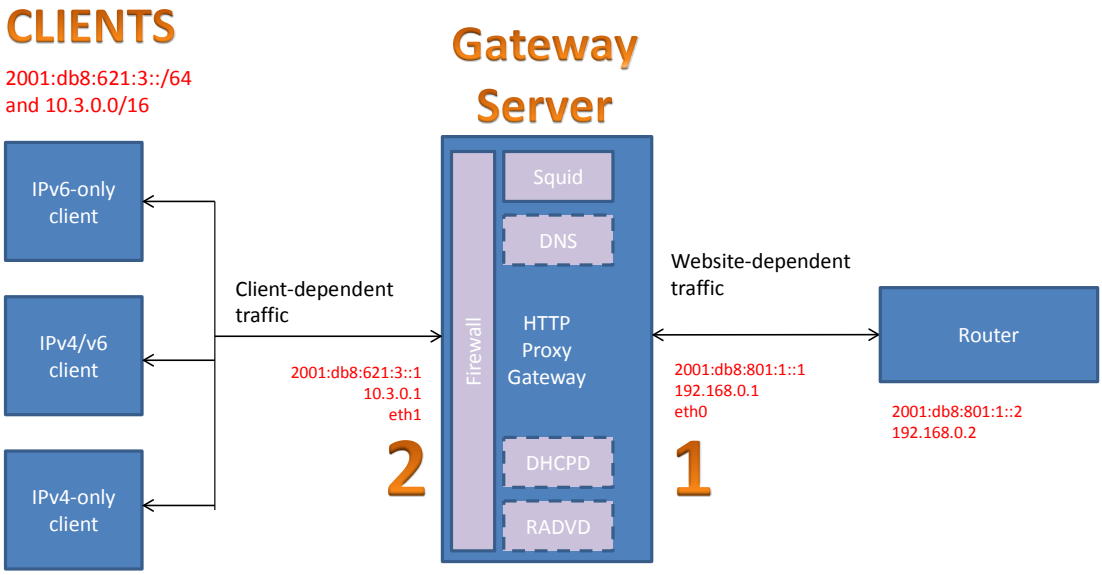

Figure 1 Server suitable for Squid Deployment, showing example IP addresses and typical services

The system shown in Figure 1, and the configurations below are illustrated using RFC1918 and RFC3849 IP addresses. Please adjust the IP addresses used to configure Squid and any associated services to match the system that you're deploying them on.

Although the system illustrates multiple external clients, the same approach can be used for clients running on the gateway server itself, but in this case their proxy server name will be set to localhost.

Installing Squid

There is a very good chance that a prebuilt Squid package will already exist for your system – it is important that you use the 3.1 stable releases rather than the older 2.X versions which did not support IPv6. If your distribution doesn't have a pre-built squid3 package then you can always build it from source. See the appendix for details on this step.

Squid runtime configuration

Squid is a very capable package and consequently has an exhaustive list of potential configuration options. The configuration described below merely provides caching-proxy access with few access restrictions, etc. The list of possible options is vast and the best place to discover potential configurations is either from the Squid website and associated documentation or by perusing the squid.conf.documented configuration file. This file will either be installed alongside the main squid.conf file if you perform an install from source or will likely be included in a shared documentation area (e.g. **`/usr/share/doc`**) for pre-built object installs. Whatever deployment you plan it is strongly suggested that you start with a simple configuration and expand it once you have successfully tested it.

Using the default configuration file (**`squid.conf`** or **`squid.conf.default`**) as a starting point, then you need to modify at least the following sections in order to configure a simplistic web-proxy role for Squid:

1. Add and/or remove localnet source addresses so that the list matches your client IPv4 and IPv6 address ranges. Based on the system shown in Figure 1 then we add entries for the served IPv4 and IPv6 subnets highlighted above:

```
acl localnet src 10.3.0.0/16
acl localnet src 2001:db8:621:3::/64
```

2. Modify the list of allowed HTTPS and HTTP safe ports, if required. The defaults are probably acceptable, at least for testing purposes.
3. Uncomment the **http_access deny to_localhost** line unless the gateway server provides HTTP proxy services to locally running processes too.
4. Adjust the Squid port listening statements. The default example is to listen on all interfaces which is probably undesirable, especially for internet bound interfaces. For example, to match the server in Figure 1 then the following statements would be required:

```
# Squid normally listens to port 3128
http_port 10.3.0.1:3128
http_port 127.0.0.1:3128
http_port [2001:db8:621:3::1]:3128
http_port [::1]:3128
```

5. Add a disk cache directory. The first number after the cache path is the size of the cache in MB (128 MB in this case). The final two figures determine the underlying cache directory structure and need not be changed.

```
cache_dir ufs /var/cache/squid3 128 16 256
```

6. Determine the upper limit of RAM for the cache to consume. This choice is dependent upon the specification of the machine acting as your gateway and the number of clients you expect to support. Note that the absolute maximum memory used by Squid can exceed this value by a small margin.

```
cache_mem 256 MB
```

7. Determine the maximum size of object that will be cached, and the maximum size of object that will be resident in memory rather than on disk. Most HTTP objects are relatively small, but you may gain significant performance advantages if frequently downloaded service-packs can be stored in the cache.

```
maximum_object_size_in_memory 8 MB
maximum_object_size 128 MB
```

8. If you're intending to analyse the cache logs then the following logformat definitions and log file locations will be useful:

```
logformat combined %>a %ui %un [%tl] "%rm %ru HTTP/%rv" %>Hs %<st
"%{Referer}>h" "%{User-Agent}>h" %Ss:%Sh
access_log /var/log/squid3/access.log combined
cache_store_log none
```

9. Some websites produce large HTTP headers, so increasing the maximum supported header size can be useful:

```
request_header_max_size 384 KB
reply_header_max_size 384 KB
```

10. The following statements suppress some of the standard Squid HTTP headers which increases your online privacy and security whilst still indicating that you are using a proxy:

```
forwarded_for off
via on
httpd_suppress_version_string on
```

The typical HTTP headers produced by this configuration would include:

```
Via: 1.1 wwwgate.example.com (squid)
X-Forwarded-For: unknown
```

11. The following statement is useful if you want to provide access to HTTP resources in your local DNS domain. This enables the use of web-addresses like http://printer instead of needing to enter the full qualified domain name, e.g. http://printer.example.com

```
append_domain .example.com
```

An example squid.conf file corresponding to the system in Figure 1 is included for reference.

Configuration of related services

Firewall

Assuming that you have a firewall active on the client-facing interface then it will be necessary to add rules to the INPUT chains to permit Squid traffic through on the appropriate TCP ports (3128 unless you chose a different port in step 4 above):

```
iptables -A INPUT –i eth1 –p tcp --dport 3128 –j ACCEPT
ip6tables -A INPUT –i eth1 –p tcp --dport 3128 –j ACCEPT
```

If your clients access Squid services using both IP version 4 and version 6 then it is necessary to open the ports for both IPv4 and IPv6 traffic in their respective firewall configurations.

Firewall rules may also be required to prevent clients directly accessing web services.

DNS Modifications

There are potentially two changes that may be made to your DNS. One impacts the forwarders that the DNS service uses and the other impacts the entries related to the Squid gateway.

The order of DNS forwarders is important if dual-stacked websites are to be reliably accessed using IPv6. It is suggested that preference be given to IPv6 accessible DNS resolvers whenever possible. If you are using a tunnel broker to provide IPv6 access then it is suggested that the broker's DNS servers are placed at the head of your list of DNS resolvers. In addition, several companies are now offering publicly-accessible IPv6 DNS servers, including Google and OpenDNS.

Although not strictly necessary to support Squid, it is suggested that you add DNS entries specifically for the HTTP gateway. Not only does this save you and your users having to remember and enter IPv6 addresses but it also allows a single DNS entry to be used as the gateway proxy irrespective of whether the client is single or dual-stacked. Based on the example in Figure 1 then the following set are suggested, not only for real-operation but also to ease testing, as will be described below:

1. add AAAA and A entries on consecutive lines to the appropriate DNS configuration file for wwwgate. If you are deploying clients using a single version of IP (e.g. continuing only IPv4 on your internal networks) then you need only include a single entry appropriate to your clients' addressing method and restrict your testing appropriately.

```
wwwgate         AAAA            2001:db8:621:3::1
                A               10.3.0.1
```

2. For testing purposes add an A record for wwwgate4 and an AAAA record for wwwgate6. These entries are not required if your clients are restricted to using a single IP version:

```
wwwgate4        A               10.3.0.1
wwwgate6        AAAA            2001:db8:621:3::1
```

   These latter entries will enable you to manually force your browser to establish either an IPv4 or IPv6 session with Squid. Once testing is complete you can remove these entries if you feel it necessary.

Starting The Squid Service

In general many of the steps described in the following sections will require root privileges. Given the wide variety of ways to achieve this in modern Linux distributions it is left up to the user to prefix individual commands with sudo, or to run them in a privileged shell.

A similar issue, applies to the management of system services, which can typically be controlled using /etc/init.d scripts, systemctl calls or service calls dependent on the distribution and version that you are using. Rather than repeat the entire list of potential commands for any given step, an example list is given below and the reader is left to determine the call appropriate for their system.

First it is necessary to attempt to start the Squid service using the appropriate service call for your system. Unfortunately, different distributions use different service management routines as well as

different names for both the service itself and the executable (some install it as squid, others as squid3). It is likely that one of the following, or a variant thereof, will enable you to start the service:

```
# /etc/init.d/squid start
```

or:

```
# systemctl start squid.service
```

or:

```
# service squid3 start
```

It is important to then check for any errors or warnings that may have appeared in /var/log/messages, /var/log/syslog or Squid's /var/log/squid3/cache.log.

One relatively frequent first-run error, is the need to build the cache directory structure prior to starting the Squid service:

```
Apr 29 10:23:23 ubuntu12 squid3: Failed to verify one of the swap
directories, Check cache.log for details.  Run 'squid -z' to create swap
directories if needed, or if running Squid for the first time.
Apr 29 10:23:23 ubuntu12 kernel: [ 315.274136] init: squid3 main process
(2034) terminated with status 1
```

This can be achieved by running Squid with the –z switch. Ensure that the directory specified in the cache_dir configuration line already exists, and is writeable by the Squid process:

```
mkdir /var/cache/squid3
chmod 777 /var/cache/squid3
```

then call the appropriate Squid executable:

```
# squid -z
```

Once the set of cache directories have been successfully built it is necessary to check the user and group id of the set of cache subdirectories and change the top level cache-directory owner to match:

```
# ls –asl /var/cache/squid3
```

shows something like the following:

```
total 72
4 drwxrwxrwx  18 root  root  4096 Apr 26 15:53 .
4 drwxr-xr-x  18 root  root  4096 Apr 26 15:53 ..
4 drwxr-x--- 258 proxy proxy 4096 Apr 26 15:53 00
4 drwxr-x--- 258 proxy proxy 4096 Apr 26 15:53 01
… through …
4 drwxr-x--- 258 proxy proxy 4096 Apr 26 15:53 0E
4 drwxr-x--- 258 proxy proxy 4096 Apr 26 15:53 0F
```

Looking at the set of generated directories, called 00 through 0F you can see the user and group id that Squid is executing as. In some distributions, e.g. Ubuntu, the squid3 service is run as the system-user and group 'proxy' and consequently the following commands would be required:

```
chown proxy:proxy /var/cache/squid3
chmod 750 /var/cache/squid3
```

Assuming that you've cleared any other configuration issues then you can attempt to restart the Squid service.

Once you are left with a running Squid process, and no warnings in the appropriate log, then you can verify that the Squid server is listening on the correct addresses and ports using lsof and grep. The grep search term should be modified if you are using a TCP port other than 3128:

```
# lsof -i -n -P |grep ':3128'
```

```
squid 14913 squid 18u  IPv4  277869 0t0 TCP 127.0.0.1:3128 (LISTEN)
squid 14913 squid 19u  IPv4  277871 0t0 TCP 10.3.0.1:3128 (LISTEN)
squid 14913 squid 21u  IPv6  277875 0t0 TCP [::1]:3128 (LISTEN)
squid 14913 squid 22u  IPv6  277877 0t0 TCP [2001:db8:621:3::1]:3128 (LISTEN)
```

Once the Squid service is successfully started then you can begin testing using a web-browser running on the gateway server. Access using the proxy is enabled by passing the proxy address (or name if you have updated your DNS) and port in to the web-browser. This can be achieved in several ways:

1. Manual entry of the address (or name) and port into the browser
2. Configuring the gateway server to use a proxy for all network HTTP traffic – which other programs such as web-browsers can then automatically use. This approach is illustrated in Figure 2.

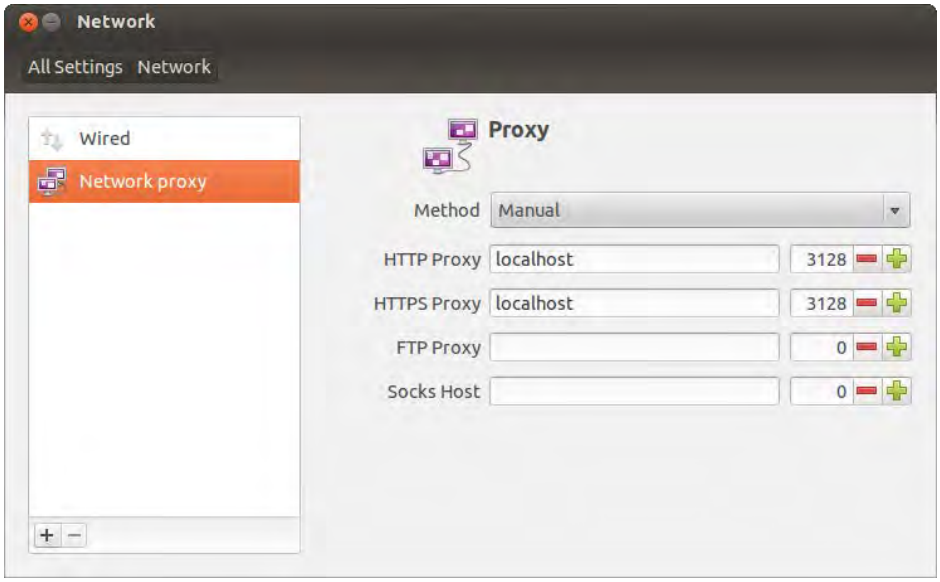

Figure 2 The Ubuntu Network Proxy Dialogue showing typical HTTP(S) entries

In some distributions, such as openSUSE, the setting of a system-wide proxy results in the standard proxy-related environment variables also being set, so that utilities like wget automatically discover the new proxy. Other distributions, such as Ubuntu, require a few additional steps to manually set these variables:

1. Edit /etc/environment
2. Add:
    a. **http_proxy="http://localhost:3128"**
    b. **https_proxy="http://localhost:3128"**
    c. **ftp_proxy=""**
    d. **no_proxy="localhost,127.0.0.1"**
3. Edit /etc/sudoers using visudo
4. Add, or merge:
    a. **Defaults env_keep="no_proxy http_proxy https_proxy ftp_proxy"**

You may decide that such changes to the proxy server environment are not worthwhile if you will rarely run a browser on the server itself. However it is useful to cover such an example in case one of your client machines is itself a Linux server.

As illustrated above the proxy variables all point to the proxy server using the same format:

service_proxy="http://server:port"

where the server can either be an IP address or a DNS name, and the port is the TCP port defined in the squid.conf file. Notice that the https_proxy variable still uses an http target when referencing the proxy server despite establishing https sessions. Note that it is usually necessary to log out and back in for the variable changes to be propagated.

Some programs, such as the online update facilities tend to automatically use the standard environment variables. Other programs, such as Firefox, may require their settings to be manually configured, as illustrated in Figure 3.

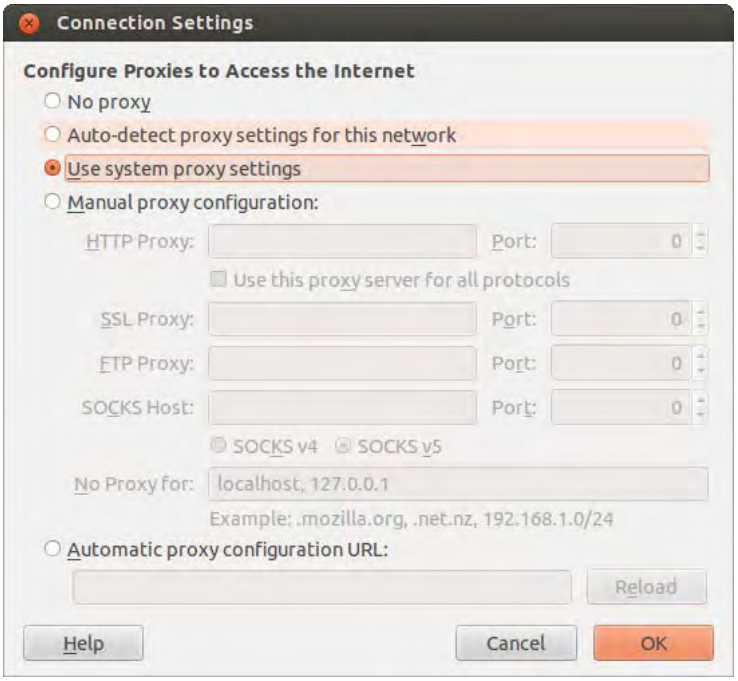

Figure 3 Firefox Connection Settings Menu, selecting the System Proxy settings

Having setup the Squid service, it is now possible to begin thorough testing.

Testing

It is recommended that you begin testing by verifying that your existing direct IPv4 and IPv6 web access is working as expected. This can be done with any web-browser or similar program running on the server gateway. The wget utility is recommended for this purpose since not only can it be run within a shell but it can also be manually forced to prefer either IPv4 or IPv6 hosts with a command line switch.

To verify existing IPv4 functionality then the following command could be used:

```
# wget -4 --no-proxy --delete-after http://www.linuxpromagazine.com
```

The -4 parameter forces an IPv4 address lookup request, the --no-proxy parameter forces wget to connect directly to the webserver rather than using any environment variables to determine which proxy to use and the - -delete-after option forces wget to delete any downloaded files which means there's no cleaning up to do afterwards. The response should appear something like the following:

```
# wget -4 --no-proxy --delete-after http://www.linuxpromagazine.com
--2012-04-30 08:44:35--  http://www.linuxpromagazine.com/
Resolving www.linuxpromagazine.com (www.linuxpromagazine.com)...
195.122.146.149
Connecting to www.linuxpromagazine.com
(www.linuxpromagazine.com)|195.122.146.149|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: `index.html'
    [ <=>
] 61,268      --.-K/s   in 0.1s
2012-04-30 08:44:36 (400 KB/s) - `index.html' saved [61268]
```

```
Removing index.html.
```

The “200 OK” response indicates that the remote web server received and understood the request and will serve the requested content. The log also shows the IP address of the server which it is communicating with - it is useful to check this address since more and more websites are becoming dual stacked and consequently without the option to force a preferred IP version you may end up accessing the wrong IP version.

Assuming this request was successful then you can attempt the same transfer towards an IPv6 enabled website, e.g.

```
# wget -6 --no-proxy --delete-after  http://ipv6.google.com
```

Again verify that the expected ‘200 OK’ response is received.

Note that if you set a preference for a protocol version which the website doesn’t advertise in its DNS entry, then you will receive a response similar to:

```
# wget -6 --no-proxy --delete-after http://www.linuxpromagazine.com
--2012-04-30 08:47:12--  http://www.linuxpromagazine.com/
Resolving www.linuxpromagazine.com (www.linuxpromagazine.com)... failed:
Name or service not known.
wget: unable to resolve host address `www.linuxpromagazine.com'
```

Using your browser of choice, or else wget, you can now check that irrespective of whether you connect to Squid using an IPv4 connection or an IPv6 one that you can browse both IPv4 and IPv6 websites. First forcing an IPv4 connection to Squid:

```
# export http_proxy=http://wwwgate4:3128

# wget --delete-after http://ipv6.google.com
--2012-04-28 15:19:50--  http://ipv6.google.com/
Resolving wwwgate4 (wwwgate4)... 10.3.0.1
Connecting to wwwgate4 (wwwgate4)|10.3.0.1|:3128... connected.
Proxy request sent, awaiting response... 200 OK

# wget --delete-after http://www.linuxpromagazine.com
--2012-04-28 15:22:22--  http://www.linuxpromagazine.com/
Resolving wwwgate4 (wwwgate4)... 10.3.0.1
Connecting to wwwgate4 (wwwgate4)|10.3.0.1|:3128... connected.
Proxy request sent, awaiting response... 200 OK
```

Figure 4 illustrates the same process using Konqueror, configured to use wwwgate4.example.com as its’ proxy server and accessing Google’s IPv6-only website:

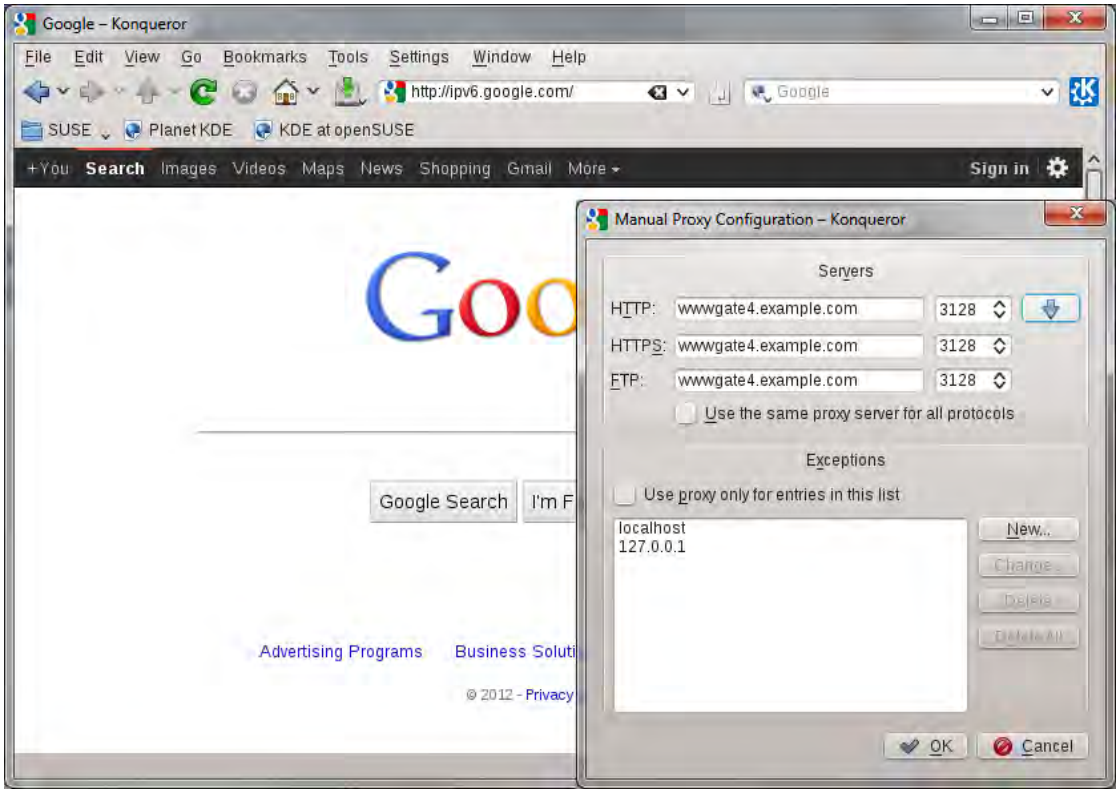

Figure 4 Konqueror, accessing Squid via IPv4, towards Google's IPv6-only site

The same tests can be repeated, but this time using an IPv6 connection to the proxy, by selecting the wwwgate6 hostname or its equivalent IPv6 address. If entering an IPv6 literal address which forms part of an URL then it must be encased in square brackets:

```
# export http_proxy=http://[2001:db8:621:3::1]:3128

# wget --delete-after http://ipv6.google.com
--2012-04-28 15:25:42--  http://ipv6.google.com/
Connecting to 2001:db8:621:3::1:3128... connected.
Proxy request sent, awaiting response... 200 OK

# wget --delete-after http://www.linuxpromagazine.com
--2012-04-28 15:25:47--  http://www.linuxpromagazine.com/
Connecting to 2001:db8:621:3::1:3128... connected.
Proxy request sent, awaiting response... 200 OK
```

Note that in the above examples I have only changed the proxy server for HTTP traffic, and not the HTTPS entry, since the websites we're testing access for are only accessed using the HTTP protocol. For client browser configuration it is normal to change the entries for both HTTP and HTTPS (sometimes labelled Secure) traffic to point to the same proxy server since many real-world websites will use both protocol types.

Assuming that these tests are successful then they can be repeated using browsers on client devices. The system can then be deployed on all clients irrespective of the IP version that they use as long as the generic wwwgate name is used as the web proxy target and your DNS server supports lookups in the same IP version.

If you use direct IP addresses as the proxy host, e.g. **`http://10.3.0.1`** instead of allocating a DNS name, then the client will be limited to connecting to the Squid proxy using the IP version that matches the IP address you specify.

Once you're satisfied that the Squid service is operating correctly then it should be configured to ensure it starts at server boot time using an appropriate service-enabling command, e.g.:

```
chkconfig squid on
```

Client configuration

The settings required for clients depend on both the user-agents (browsers) and operating systems in use. In general the following points should be noted:

1. Standalone clients will generally use either a manual/static proxy configuration, or potentially an auto-detect approach if you have deployed a WPAD enabled system. System-wide proxy settings will only work if you have previously configured the client machine to use a system proxy.
2. Typically both the HTTP and HTTPS entries will point to the same server and port – this is the case for the system illustrated in Figure 1.
3. IPv6 literal addresses, e.g. 2001:db8:621:3::1, entered as part of an URL must be encased in square brackets, as illustrated in Figure 5.
4. If the proxy entry form supports a separate server and port field then it is not usually necessary to include a protocol prefix (e.g. http://). However some browsers, e.g. Konqueror, definitely require the prefix when specifying IPv6 literal addresses.

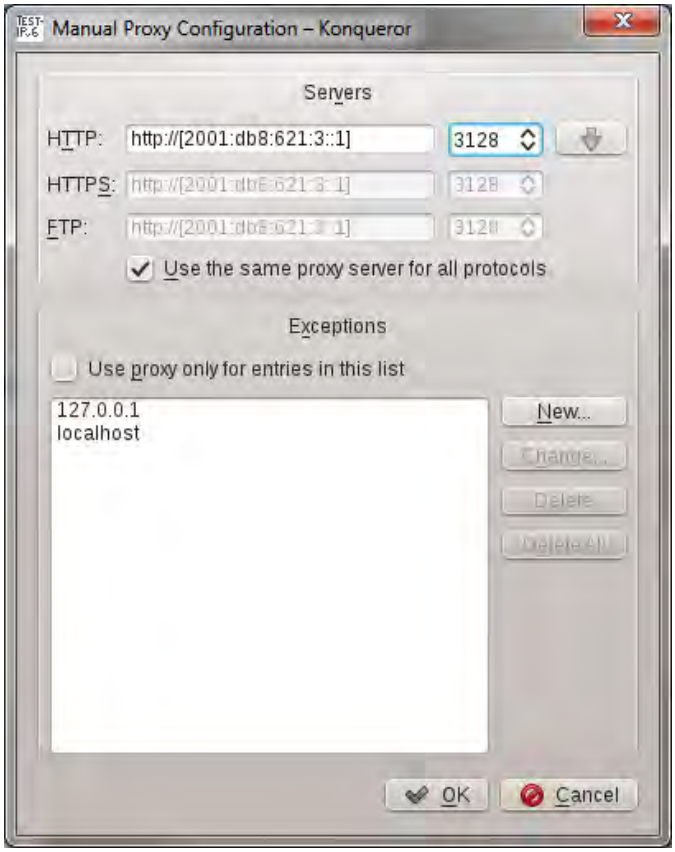

Figure 5 Konqueror entries for an IPv6 Literal Address

5. If the proxy entry specifies an URL, as is often the case for system level proxies, then the usual format is: http://server:port as illustrated in Figure 6.

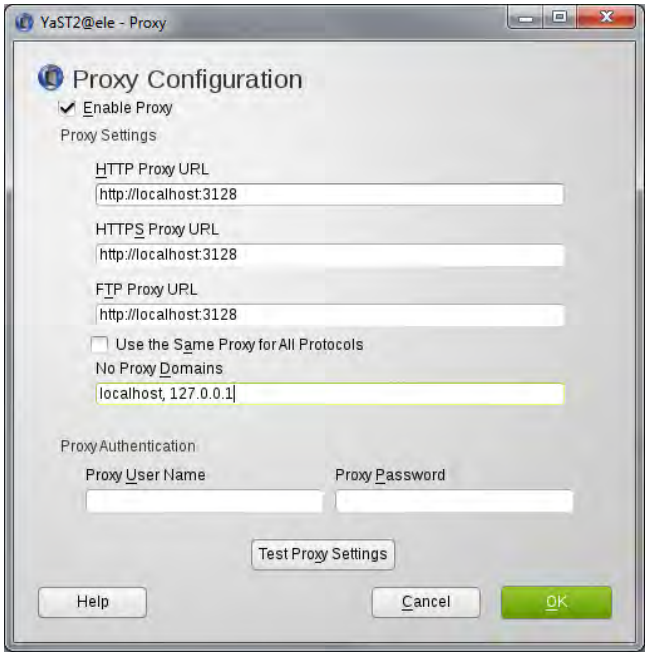

Figure 6 openSUSE System Proxy Configuration

6. It is usual to bypass the proxy when accessing the localhost, or IPv4 address 127.0.0.1 since these refer to web services running on the client itself which are usually not accessible by, and therefore available via, the Squid proxy. As IPv6 services become more prevalent then it may be necessary to also include the IPv6 localhost equivalent, ::1 on IPv6 enabled clients.

If you're in doubt whether a client is successfully using your Squid proxy for web access then simply tail the Squid access log and open a web session on the client:

```
# tailf /var/log/squid3/access.log
```

Each line details a single HTTP transaction and will include the client address, the URI of each requested item, the result of that transfer attempt and an indication as to whether Squid provided the data from its local cache or had to fetch it from the remote server.

Client Complications

One type of client in particular can cause complications. The majority of Android devices, at least those running OS versions 2.x and earlier, require an IPv4 address allocating over Wi-Fi prior to them accepting an IPv6 address via SLAAC. Whilst this in itself doesn't prevent the use of Squid, their lack of support for an HTTP proxy is somewhat more problematic. For some users who have already rooted their device then there are multiple proxy-setting apps available in the Android marketplace. However, for those with early unmodified Android phones then unfortunately this approach may not be applicable.

Known to work with …

This IP version translation approach is known to work well with a wide mixture of traditional PC clients as well as many Wifi-enabled standalone client devices, including:

- IPv4-only clients – such as the original Apple iPod Touch and Sony PlayStation Portable
- IPv4/IPv6 clients – such as Apple iPhone 3GS onwards and recent Android devices
- IPv6-only clients – supported by various OS's including Linux and iOS

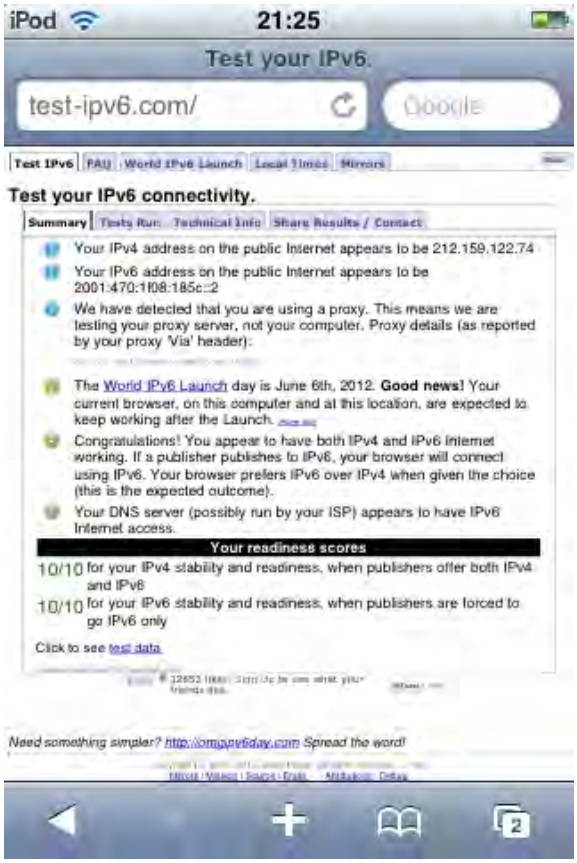

Figure 7 An original IPv4-only iPod Touch showing its surprisingly complete IPv6 compliance!

Other things you can do with Squid

In addition to Internet Protocol version translation, Squid supports a wide variety of configuration options and can be used to strictly control client access to the internet. This can be useful in several scenarios:

- preventing access to undesirable websites
- limiting access to certain websites during particular times of the day
- prevent the download of certain file types
- prevent access to adware/tracking sites to increase your online privacy
- authenticating users before web access is allowed
- limiting FTP access

The Squid website covers configurations for many of these scenarios, and is the best starting point for anyone wishing to investigate these options.

Potentially the **`access.log`** file which Squid creates can grow very quickly, particularly if you have many clients. It is suggested that you ensure logrotate is configured to check the size and age of the file and rotate to a new file every so often. Prebuilt packages often include a generic logrotate configuration, but it is worth checking that this contains suitable size/age parameters.

The **access.log** file details each and every transaction passing through Squid, and consequently contains very useful information for system administrators. There are many packages available for analysing, and summarising Squid log files and many of these are again referenced on the Squid website: http://www.squid-cache.org/Misc/log-analysis.html

If your users find proxy configuration difficult then it is possible, for some browsers at least, to support automatic proxy configuration. There are many online articles related to the Web Proxy Auto Discovery (WPAD) protocol which will provide a good overview of the additional steps required.

APPENDIX

Building From Source

Assuming that you have the necessary build environment installed then the steps required to build and install squid manually are to fetch the source, set your compiler environment variables appropriately and then configure and make Squid.

1. Fetching the source:

```
cd /usr/local/src/squid31
wget http://www.squid-cache.org/Versions/v3/3.1/squid-3.1.19.tar.gz
tar zxvf squid-3.1.19.tar.gz
cd squid-3.1.19
```

2. Setting your compiler flags appropriately. If you're running on an older 32-bit distribution then it may be necessary to force a processor device architecture later than or equal to i486 – in order that the appropriate instruction set is available. If you fail to do this then you are likely to encounter build errors related to undefined references to '__sync_fetch_and_add_4' and similar.

```
export CFLAGS='-march=i586 -mtune=i686 -fmessage-length=0 -O2 -Wall
-D_FORTIFY_SOURCE=2 -fstack-protector -funwind-tables -fasynchronous-
unwind-tables -fPIE -fPIC -fno-strict-aliasing'
export CXXFLAGS='-march=i586 -mtune=i686 -fmessage-length=0 -O2 -
Wall -D_FORTIFY_SOURCE=2 -fstack-protector -funwind-tables -fasynchronous-
unwind-tables -fPIC -fno-strict-aliasing'
export LDFLAGS='-pie'
```

3. Configuring the squid build. The options that you need will obviously depend on the way you wish to configure Squid. A minimum configuration would just require the build prefix to be configured. If you later decide to enable more complex features then you can simply reconfigure the build appropriately, rebuild, stop any running Squid sessions and then re-install the updated build.

```
./configure --prefix=/usr/local
make && make install
```

If you're unsure as to the options which your Linux distribution usually specifies, in particular which paths/prefixes to use, then you can temporarily install a pre-built version and use:

```
/path/to/existing/squid –v
```

to find the configuration options.

For reference, a more complete openSUSE build usually configures the following options:

```
./configure --prefix=/usr --sysconfdir=/etc/squid --
bindir=/usr/sbin --sbindir=/usr/sbin --localstatedir=/var --
libexecdir=/usr/sbin --datadir=/usr/share/squid --libdir=/usr/lib --with-dl
--sharedstatedir=/var/squid --enable-storeio=aufs,diskd,ufs --enable-disk-
```

```
io=AIO,Blocking,DiskDaemon,DiskThreads --enable-removal-policies=heap,lru --enable-icmp --enable-delay-pools --enable-esi --enable-icap-client --enable-useragent-log --enable-referer-log --enable-kill-parent-hack --enable-snmp --enable-arp-acl --enable-htcp --enable-ssl --enable-forw-via-db --enable-cache-digests --enable-poll --enable-linux-netfilter --with-large-files --enable-underscores --enable-auth --enable-basic-auth-helpers=DB,LDAP,MSNT,NCSA,PAM,POP3,SASL,SMB,YP,getpwnam,multi-domain-NTLM,squid_radius_auth --enable-ntlm-auth-helpers=SMB,no_check,fakeauth --enable-negotiate-auth-helpers=squid_kerb_auth --enable-digest-auth-helpers=eDirectory,ldap,password --enable-external-acl-helpers=ip_user,ldap_group,session,unix_group,wbinfo_group --enable-ntlm-fail-open --enable-stacktraces --enable-x-accelerator-vary --with-default-user=squid
```

[i] http://www.worldipv6launch.org/ World IPv6 Launch Day, 6th June 2012

[ii] http://www.squid-cache.org, "Optimising Web Delivery"