Bandwidth Limiting HOWTO

Tomasz Chmielewski

tch@metalab.unc.edu

Revision History

Revision 0.9	2001−11−20	Revised by: tc

This document describes how to set up your Linux server to limit download bandwidth or incoming traffic and how to use your internet link more efficiently.

Table of Contents

1. Introduction

The purpose of this guide is to provide an easy solution for limiting incoming traffic, thus preventing our LAN users from consuming all the bandwidth of our internet link.

This is useful when our internet link is slow or our LAN users download tons of mp3s and the newest Linux distro's *.iso files.

1.1. New versions of this document

You can always view the latest version of this document on the World Wide Web at the URL http://www.linuxdoc.org.

New versions of this document will also be uploaded to various Linux WWW and FTP sites, including the LDP home page at http://www.linuxdoc.org.

1.2. Disclaimer

Neither the author nor the distributors, or any other contributor of this HOWTO are in any way responsible for physical, financial, moral or any other type of damage incurred by following the suggestions in this text.

1.3. Copyright and License

1.4. Feedback and corrections

If you have questions or comments about this document, please feel free to mail Tomasz Chmielewski at *tch@metalab.unc.edu*. I welcome any suggestions or criticisms. If you find a mistake or a typo in this document (and you will find a lot of them, as English is not my native language), please let me know so I can correct it in the next version. Thanks.

1.5. Thanks

I would like to thank Ami M. Echeverri lula@pollywog.com who helped me to convert the HOWTO into SGML format and corrected some mistakes. I also want to thank Ryszard Prosowicz prosowicz@poczta.fm for useful suggestions.

2. Before We Start

Let's imagine the following situation:

- We have 115,2 kbits/s ppp (modem) internet link (115,2/10 = 11,5 kbytes/s). Note: with eth connections (network card) we would divide 115,2 by 8; with ppp we divide by 10, because of start/stop bits (8 + 1 + 1 = 10).
- We have some LAN stations and their users are doing bulk downloads all the time.
- We want web pages to open fast, no matter how many dowloads are happening.
- Our internet interface is **ppp0**.
- Our LAN interface is **eth0**.
- Our network is 192.168.1.0/24

2.1. What do we need

Believe it or not, shaping the incoming traffic is an easy task and you don't have to read tons of books about routing or queuing algorithms.

To make it work, we need at least Squid proxy; if we want to fine tune it, we will have to get familiar with ipchains or iptables and CBQ.

To test our efforts, we can install IPTraf.

2.2. How does it work?

Squid is probably the most advanced HTTP proxy server available for Linux. It can help us save bandwidth in two ways:

- The first is a main characteristic of proxy servers –– they keep downloaded web pages, pictures, and other objects in memory or on a disk. So, if two people are requesting the same web page, it isn't downloaded from the internet, but from the local proxy.
- Apart from normal caching, Squid has a special feature called delay pools. Thanks to delay pools, it is possible to limit internet traffic in a reasonable way, depending on so–called 'magic words', existing in any given URL. For example, a magic word could be '.mp3', '.exe' or '.avi', etc. Any distinct part of a URL (such as .avi) can be defined as a magic word.

With that, we can tell the Squid to download these kinds of files at a specified speed (in our example, it will be about 5 kbytes/s). If our LAN users download files at the same time, they will be downloaded at about 5 kbytes/s altogether, leaving remaining bandwidth for web pages, e–mail, news, irc, etc.

Of course, the Internet is not only used for downloading files via web pages (http or ftp). Later on, we will deal with limiting bandwidth for Napster, Realaudio, and other possibilities.

3. Installing and Configuring Necessary Software

Here, I will explain how to install the necessary software so that we can limit and test the bandwidth usage.

3.1. Installing Squid with the delay pools feature

As I mentioned before, Squid has a feature called delay pools, which allows us to control download bandwidth. Unfortunately, in most distributions, Squid is shipped without that feature.

So if you have Squid already installed, I must disappoint you −− you need to uninstall it and do it once again with delay pools enabled in the way I explain below.

1. To get maximum performance from our Squid proxy, it's best to create a separate partition for its cache, called /cache/. Its size should be about 300 megabytes, depending on our needs.

   If you don't know how to make a separate partition, you can create the /cache/ directory on a main partition, but Squid performance can suffer a bit.

2. We add a safe 'squid' user:

   ```
   # useradd −d /cache/ −r −s /dev/null squid >/dev/null 2>&1
   ```

   No one can log in as squid, including root.

3. We download Squid sources from http://www.squid−cache.org

   When I was writing this HOWTO, the latest version was Squid 2.4 stable 1:

   http://www.squid−cache.org/Versions/v2/2.4/squid−2.4.STABLE1−src.tar.gz

4. We unpack everything to `/var/tmp`:
5. `# tar xzpf squid−2.4.STABLE1−src.tar.gz`
6. We compile and install Squid (everthing is in one line):

   ```
   # ./configure −−prefix=/opt/squid −−exec−prefix=/opt/squid
   −−enable−delay−pools −−enable−cache−digests −−enable−poll
   −−disable−ident−lookups −−enable−truncate −−enable−removal−policies

   # make all

   # make install
   ```

3.2. Configuring Squid to use the delay pools feature

1. Configure our squid.conf file (located under /opt/squid/etc/squid.conf):

   ```
   #squid.conf
   #Every option in this file is very well documented in the original squid.conf file
   ```

```
#and on http://www.visolve.com/squidman/Configuration%20Guide.html

#
#The ports our Squid will listen on.
http_port 8080
icp_port 3130
#cgi-bins will not be cached.
acl QUERY urlpath_regex cgi-bin \?
no_cache deny QUERY
#Memory the Squid will use. Well, Squid will use far more than that.
cache_mem 16 MB
#250 means that Squid will use 250 megabytes of disk space.
cache_dir ufs /cache 250 16 256

#Places where Squid's logs will go to.
cache_log /var/log/squid/cache.log
cache_access_log /var/log/squid/access.log
cache_store_log /var/log/squid/store.log
cache_swap_log /var/log/squid/swap.log
#How many times to rotate the logs before deleting them.
#See the FAQ for more info.
logfile_rotate 10

redirect_rewrites_host_header off
cache_replacement_policy GDSF
acl localnet src 192.168.1.0/255.255.255.0
acl localhost src 127.0.0.1/255.255.255.255
acl Safe_ports port 80 443 210 119 70 20 21 1025-65535
acl CONNECT method CONNECT
acl all src 0.0.0.0/0.0.0.0
http_access allow localnet
http_access allow localhost
http_access deny !Safe_ports
http_access deny CONNECT
http_access deny all
maximum_object_size 3000 KB
store_avg_object_size 50 KB

#Set these if you want your proxy to work in a transparent way.
#Transparent proxy means you generally don't have to configure all
#your client's browsers, but hase some drawbacks too.
#Leaving these uncommented won't do any harm.
httpd_accel_host virtual
httpd_accel_port 80
httpd_accel_with_proxy on
httpd_accel_uses_host_header on

#all our LAN users will be seen by external web servers
#as if they all used Mozilla on Linux. :)
anonymize_headers deny User-Agent
fake_user_agent Mozilla/5.0 (X11; U; Linux i686; en-US; rv:0.9.6+) Gecko/20011122

#To make our connection even faster, we put two lines similar
#to the ones below. They will point a parent proxy server our own Squid
#will use. Don't forget to change the server to the one that will
#be fastest for you!
#Measure pings, traceroutes and so on.
#Make sure that http and icp ports are correct.

#Uncomment lines beginning with "cache_peer" if necessary.
#This is the proxy you are going to use for all connections...
#cache_peer w3cache.icm.edu.pl parent 8080 3130 no-digest default
```

```
#...except for the connections to addresses and IPs beginning with "!".
#It's a good idea not to use a higher
#cache_peer_domain w3cache.icm.edu.pl !.pl !7thguard.net !192.168.1.1

#This is useful when we want to use the Cache Manager.
#Copy cachemgr.cgi to cgi-bin of your www server.
#You can reach it then via a web browser typing
#the address http://your-web-server/cgi-bin/cachemgr.cgi
cache_mgr your@email
cachemgr_passwd secret_password all

#This is a name of a user our Squid will work as.
cache_effective_user squid
cache_effective_group squid

log_icp_queries off
buffered_logs on


#####DELAY POOLS
#This is the most important part for shaping incoming traffic with Squid
#For detailed description see squid.conf file or docs at http://www.squid-cache.org

#We don't want to limit downloads on our local network.
acl magic_words1 url_regex -i 192.168

#We want to limit downloads of these type of files
#Put this all in one line
acl magic_words2 url_regex -i ftp .exe .mp3 .vqf .tar.gz .gz .rpm .zip .rar .avi .mpeg .mpe .mpg .qt
.ram .rm .iso .raw .wav .mov
#We don't block .html, .gif, .jpg and similar files, because they
#generally don't consume much bandwidth

#We want to limit bandwidth during the day, and allow
#full bandwidth during the night
#Caution! with the acl below your downloads are likely to break
#at 23:59. Read the FAQ in this bandwidth if you want to avoid it.
acl day time 09:00-23:59

#We have two different delay_pools
#View Squid documentation to get familiar
#with delay_pools and delay_class.
delay_pools 2

#First delay pool
#We don't want to delay our local traffic.
#There are three pool classes; here we will deal only with the second.
#First delay class (1) of second type (2).
delay_class 1 2

#-1/-1 mean that there are no limits.
delay_parameters 1 -1/-1 -1/-1

#magic_words1: 192.168 we have set before
delay_access 1 allow magic_words1


#Second delay pool.
#we want to delay downloading files mentioned in magic_words2.
#Second delay class (2) of second type (2).
delay_class 2 2
```

```
#The numbers here are values in bytes;
#we must remember that Squid doesn't consider start/stop bits
#5000/150000 are values for the whole network
#5000/120000 are values for the single IP
#after downloaded files exceed about 150000 bytes,
#(or even twice or three times as much)
#they will continue to download at about 5000 bytes/s

delay_parameters 2 5000/150000 5000/120000
#We have set day to 09:00-23:59 before.
delay_access 2 allow day
delay_access 2 deny !day
delay_access 2 allow magic_words2


#EOF
```

OK, when we have configured everything, we must make sure everything under `/opt/squid` and `/cache` directories belongs to user 'squid'.

mkdir /var/log/squid/

chown squid:squid /var/log/squid/

chmod 770 /var/log/squid/

chown –R squid:squid /opt/squid/

chown –R squid:squid /cache/

Now everything is ready to run Squid. When we do it for the first time, we have to create its cache directories:

/opt/squid/bin/squid –z

We run Squid and check if everything is working. A good tool to do that is IPTraf; you can find it on http://freshmeat.net. Make sure you have set the appropriate proxy in your web browsers (192.168.1.1, port 8080 in our example):

/opt/squid/bin/squid

If everything is working, we add `/opt/squid/bin/squid` line to the end of our initializing scripts. Usually, it can be `/etc/rc.d/rc.local`.

Other helpful options in Squid may be:

/opt/squid/bin/squid –k reconfigure (it reconfigures Squid if we made any changes in its squid.conf file)

/opt/squid/bin/squid –help :) self–explanatory

You can also copy `cachemgr.cgi` to the cgi–bin directory of your WWW server, to make use of a useful Cache Manager.

3.3. Solving remaining problems

OK, we have installed Squid and configured it to use delay pools. I bet nobody wants to be restricted, especially our clever LAN users. They will likely try to avoid our limitations, just to download their favourite mp3s a little faster (and thus causing your headache).

I assume that you use IP−masquerade on your LAN so that your users could use IRC, ICQ, e−mail, etc. That's OK, but we must make sure that our LAN users will use our delay pooled Squid to access web pages and use `ftp`.

We can solve most of these problems by using `ipchains` (Linux 2.2.x kernels) or `iptables` (Linux 2.4.x kernels).

3.3.1. Linux 2.2.x kernels (ipchains)

We must make sure that nobody will try to cheat and use a proxy server other than ours. Public proxies usually run on 3128 and 8080 ports:

/sbin/ipchains −A input −s 192.168.1.1/24 −d ! 192.168.1.1 3128 −p TCP −j REJECT

/sbin/ipchains −A input −s 192.168.1.1/24 −d ! 192.168.1.1 8080 −p TCP −j REJECT

We must also make sure that nobody will try to cheat and connect to the internet directly (IP−masquerade) to download web pages:

/sbin/ipchains −A input −s 192.168.1.1/24 −d ! 192.168.1.1 80 −p TCP −j REDIRECT 8080

If everything is working, we add these lines to the end of our initializing scripts. Usually, it can be `/etc/rc.d/rc.local`.

We might think to block `ftp` traffic (ports 20 and 21) to force our LAN users to use Squid, but it's not a good idea for at least two reasons:

- Squid is a http proxy with `ftp` support, not a real `ftp` proxy. It can download from `ftp`, it can also upload to some `ftp`, but it can't delete/change name of files on remote `ftp` servers.

  When we block ports 20 and 21, we won't be able to delete/change name of files on remote `ftp` servers.

- IE5.5 has a bug −− it doesn't use a proxy to retrieve the `ftp` directory. Instead it connects directly via IP−masquerade.

  When we block ports 20 and 21, we won't be able to browse through `ftp` directories, using IE5.5.

So, we will block excessive `ftp` downloads using other methods. We will deal with it in chapter 4.

3.3.2. Linux 2.4.x kernels (iptables)

We must make sure that nobody will try to cheat and use a proxy server other than ours. Public proxies usually run on 3128 and 8080 ports:

/sbin/iptables −A FORWARD −s 192.168.1.1/24 −d ! 192.168.1.1 −−dport 3128 −p TCP −j DROP

/sbin/iptables −A FORWARD −s 192.168.1.1/24 −d ! 192.168.1.1 −−dport 8080 −p TCP −j DROP

We must also make sure that nobody will try to cheat and connect to the internet directly (IP−masquerade) to download web pages:

/sbin/iptables −t nat −A PREROUTING −i eth0 −p tcp −−dport 80 −j REDIRECT −−to−port 8080

If everything is working, we add these lines to the end of our initializing scripts. Usually, it can be `/etc/rc.d/rc.local`.

We might think to block `ftp` traffic (ports 20 and 21) to force our LAN users to use Squid, but it's not a good idea for at least two reasons:

- Squid is a http proxy with `ftp` support, not a real `ftp` proxy. It can download from `ftp`, it can also upload to some `ftp`, but it can't delete/change name of files on remote `ftp` servers.

  When we block ports 20 and 21, we won't be able to delete/change name of files on remote `ftp` servers.

- IE5.5 has a bug −− it doesn't use a proxy to retrieve the `ftp` directory. Instead it connects directly via IP−masquerade.

  When we block ports 20 and 21, our LAN users won't be able to browse through `ftp` directories, using IE5.5.

So, we will block excessive `ftp` downloads using other methods. We will deal with it in chapter 4.

4. Dealing with Other Bandwidth−consuming Protocols Using CBQ

We must remember that our LAN users can spoil our efforts from chapter 3, if they use Napster, Kazaa or Realaudio. We must also remember that we didn't block `ftp` traffic in section 3.3.

We will achieve it in a different way −− not by limiting downloading directly, but rather, indirectly. If our internet device is `ppp0` and LAN device is `eth0`, we will limit outgoing traffic on interface `eth0`, and thus, limit incoming traffic to `ppp0`.

To do it, we will get familiar with CBQ and `cbq.init` script. You can obtain it from ftp://ftp.equinox.gu.net/pub/linux/cbq/. Download `cbq.init−v0.6.2` and put it in `/etc/rc.d/`.

You will also need `iproute2` installed. It comes with every Linux distribution.

Now look in your `/etc/sysconfig/cbq/` directory. There, you should have an example file, which should work with `cbq.init`. If it isn't there, you probably don't have it compiled in your kernel nor it isn't present as modules. Well, in any case, just make that directory, put example files provided below, and see if it'd work for you.

4.1. FTP

In chapter 3, we didn't block ftp for two reasons −− so that we could do uploads, and so that users with buggy IE5.5 could browse through `ftp` directories. In all, our web browsers and `ftp` programs should make downloads via our Squid proxy and `ftp` uploads/renaming/deleting should be made via IP−masquerade.

We create a file called `cbq−10.ftp−network` in the `/etc/sysconfig/cbq/` directory:

touch /etc/sysconfig/cbq/cbq−10.ftp−network

We insert the following lines into it:

```
DEVICE=eth0,10Mbit,1Mbit
RATE=15Kbit
WEIGHT=1Kbit
PRIO=5
RULE=:20,192.168.1.0/24
RULE=:21,192.168.1.0/24
```

You will find the description of thses lines in `cbq.init−v0.6.2` file.

When you start `/etc/rc.d/cbq.init−v0.6.2` script, it will read your configuration, which is placed in `/etc/sysconfig/cbq/`:

/etc/rc.d/cbq.init−v0.6.2 start

If everything is working, we add `/etc/rc.d/cbq.init−v0.6.2 start` to the end of your initializing scripts. Usually, it can be `/etc/rc.d/rc.local`.

Thanks to this command, your server will not send `ftp` data through `eth0` faster than about 15kbits/s, and thus will not download `ftp` data from the internet faster than 15kbits/s.Your LAN users will see that it's more efficient to use Squid proxy for doing `ftp` downloads. They will be also able to browse `ftp` directories using their buggy IE5.5.

There is also another bug in IE5.5 − when you right click on a file in a `ftp` directory then select 'Copy To Folder', the file is downloaded not through proxy, but directly through IP−masquerade, thus omitting Squid with delay pools.

4.2. Napster, Realaudio, Windows Media and other issues

Here, the idea is the same as with `ftp`; we just add another port and set a different speed.

We create file called `cbq−50.napster−network` in the `/etc/sysconfig/cbq/` directory:

touch /etc/sysconfig/cbq/cbq−50.napsterandlive

Put these lines into that file:

```
DEVICE=eth0,10Mbit,1Mbit
RATE=35Kbit
WEIGHT=3Kbit
PRIO=5
#Windows Media Player.
RULE=:1755,192.168.1.0/24
#Real Player uses TCP port 554, for UDP it uses different ports,
#but generally RealAudio in UDP doesn't consume much bandwidth.
RULE=:554,192.168.1.0/24
RULE=:7070,192.169.1.0/24
#Napster uses ports 6699 and 6700, maybe some other?
RULE=:6699,192.168.1.0/24
RULE=:6700,192.168.1.0/24
#Audiogalaxy uses ports from 41000 to as high as probably 41900,
#there are many of them, so keep in mind I didn't list all of
#them here. Repeating 900 nearly the same lines would be of course
#pointless. We will simply cut out ports 410031-41900 using
#ipchains or iptables.
RULE=:41000,192.168.1.0/24
RULE=:41001,192.168.1.0/24
#continue from 41001 to 41030
RULE=:41030,192.168.1.0/24
#Some clever users can connect to SOCKS servers when using Napster,
#Audiogalaxy etc.; it's also a good idea to do so
#when you run your own SOCKS proxy
RULE=:1080,192.168.1.0/24
#Add any other ports you want; you can easily check and track
#ports that programs use with IPTraf
#RULE=:port,192.168.1.0/24
```

Don't forget to cut out remaining Audiogalaxy ports (41031−41900), using ipchains (kernels 2.2.x or iptables (kernels 2.4.x).

Kernels 2.2.x.

/sbin/ipchains −A input −s 192.168.1.1/24 −d ! 192.168.1.1 41031:41900 −p TCP −j REJECT

Kernels 2.4.x.

/sbin/iptables −A FORWARD −s 192.168.1.1/24 −d ! 192.168.1.1 −−dport 41031:41900 −p TCP −j REJECT

Don't forget to add a proper line to your initializing scripts.

5. Frequently Asked Questions

5.1. Is it possible to limit bandwidth on a per−user basis with delay pools?

Yes. Look inside the original `squid.conf` file and check the Squid documentation on http://www.squid−cache.org

5.2. How do I make wget work with Squid?

It's simple. Create a file called `.wgetrc` and put it in your home directory. Insert the following lines in it and that's it!

```
HTTP_PROXY=192.168.1.1:8080
FTP_PROXY=192.168.1.1:8080
```

You can make it work globally for all users, type `man wget` to learn how.

5.3. I set up my own SOCKS server listening on port 1080, and now I'm not able to connect to any irc server.

There can be two issues here.

One is when your SOCKS proxy is open relay, that means everyone can use it from any place in the world. It is a security issue and you should check your SOCKS proxy configuration again − generally irc servers don't allow open relay SOCKS servers to connect to them.

If you are sure your SOCKS server isn't open relay, you may be still disallowed to connect to some of the irc servers − it's because mostly they just check if SOCKS server is running on port 1080 of a client that is connecting. In that case just reconfigure your SOCKS to work on a different port. You will also have to reconfigure your LAN software to use a proper SOCKS server and port.

5.4. I don't like when Kazaa or Audiogalaxy is filling up all my upload bandwidth.

Indeed that can be painful, but it's simple to be solved.

Create a file called for example `/etc/sysconfig/cbq/cbq−15.ppp`.

Insert the following lines into it, and Kazaa or Audiogalaxy will upload not faster than about 15 kbits/s. I assume that your outgoing internet interface is ppp0.

```
DEVICE=ppp0,115Kbit,11Kbit
```

```
RATE=15Kbit
WEIGHT=2Kbit
PRIO=5
TIME=01:00-07:59;110Kbit/11Kbit
RULE=,:21
RULE=,213.25.25.101
RULE=,:1214
RULE=,:41000
RULE=,:41001
#And so on till :41030
RULE=,:41030
```

5.5. My outgoing mail server is eating up all my bandwidth.

You can limit your SMTP, Postfix, Sendmail, or whatever, in a way similar to the question above. Just change or add one rule:

```
RULE=,:25
```

Moreover, if you have an SMTP server, you can force your local LAN users to use it, even though they have set up their own SMTP servers to smtp.some.server! We'll do it in a transparent way we did before with Squid.

5.6. Can I limit my own FTP or WWW server in a manner similar it is shown in the question above?

Generally you can, but usually these servers have got their own bandwidth limiting configurations, so you will probably want to look into their documentation.

2.2.x Kernels

/sbin/ipchains −A input −s 192.168.1.1/24 −d ! 192.168.1.1 25 −p TCP −j REDIRECT 25

2.4.x Kernels

/sbin/iptables −t nat −A PREROUTING −i eth0 −p tcp −−dport 25 −j REDIRECT −−to−port 25

Don't forget to add a proper line to your initializing scripts.

5.7. Is it possible to limit bandwidth on a per−user basis with cbq.init script?

Yes. Look inside this script; there are some examples.

5.8. Whenever I start `cbq.init`, it says `sch_cbq` is missing.

Probably you don't have CBQ as modules in your system. If you have compiled CBQ into your kernel, comment out the following lines in your `cbq.init-v0.6.2` script.

```
### If you have cbq, tbf and u32 compiled into kernel, comment it out
#for module in sch_cbq sch_tbf sch_sfq sch_prio cls_u32; do
#        if ! modprobe $module; then
#                echo "**CBQ: could not load module $module"
#                exit
#        fi
#done
```

5.9. CBQ sometimes doesn't work for no reason.

Generally it shouldn't occur. Sometimes, you can observe mass downloads, though you think you have blocked all ports Napster or Audiogalaxy uses. Well, there is always one more port open for mass downloads. To find it, you can use IPTraf. As there can be possibly thousands of such ports, it can be really hard task for you. To make it easier, you can consider running your own SOCKS proxy – Napster, Audiogalaxy and many programs can use SOCKS proxies, so it's much easier to deal with just one port, than to do so with thousands of possibilites (standard SOCKS port is 1080, if you run your own SOCKS proxy server, you will be able to set it up differently, or run multiple instances of SOCKS proxy listening on different ports). Don't forget to close all ports for traffic, and leave open ports like 25 and 110 (SMTP and POP3), and other you think might be useful. You will find a link to awesome Nylon socks proxy server at the end of this HOWTO.

5.10. Delay pools are stupid; why can't I download something at full speed when the network is used only by me?

Unfortunately, you can't do much about it.

The only thing you can do is to use **cron** and reconfigure it, for example, at 1.00 am, so that Squid won't use delay pools, then reconfigure it again, let's say at 7.30 am, to use delay pools.

To do this, create two separate config files, called for example `squid.conf-day` and `squid.conf-night`, and put them into `/opt/squid/etc/`.

`squid.conf-day` would be the exact copy of a config we created earlier

`squid.conf-night`, on the contrary, would not have any delay pool lines, so all you have to do is to comment them out.

Next thing you have to do is to set up `/etc/crontab` entries correctly.

Edit `/etc/crontab` and put the following lines there:

```
#SQUID - night and day config change
```

```
01 9 * * * root /bin/cp -f /opt/squid/etc/squid.conf-day /opt/squid/etc/squid.conf; /opt/squid/bin/squid -k reconfigure
59 23 * * * root /bin/cp -f /opt/squid/etc/squid.conf-night /opt/squid/etc/squid.conf; /opt/squid/bin/squid -k reconfigure
```

5.11. My downloads break at 23:59 with "acl day time 09:00−23:59" in squid.conf. Can I do something about it?

You can achieve by removing that acl from your squid.conf, and "delay_access 2 allow dzien delay_access 2 deny !dzien" as well.

Then try to do it with **cron** as in the question above.

5.12. Squid's logs grow and grow very fast, what can I do about it?

Indeed, the more users you have, the more − sometimes useful − information will be logged.

The best way to eradicate it would be to use **logrotate**, but you'd have to do a little trick to make it work with Squid: proper **cron** and **logrotate** entries.

`/etc/crontab` entries:

```
#SQUID - logrotate
01 4 * * * root /opt/squid/bin/squid -k rotate; /usr/sbin/logrotate /etc/logrotate.conf; /bin/rm -f /var/log/squid/*.log.0
```

Here we have caused **logrotate** to start daily at 04:01 am, so remove any remaining **logrotate** starting points, for example from `/etc/cron.daily/`.

`/etc/logrotate.d/syslog` entries:

```
#SQUID logrotate - will keep logs for 40 days
/var/log/squid/*.log.0 {
rotate 40
compress
daily
postrotate
/usr/bin/killall -HUP syslogd
endscript
}
```

5.13. CBQ is stupid; why can't I download something at full speed when the network is used only be me?

Lucky you, it's possible!

There are to ways to achieve it.

The first is the easy one, similar to the solution we've made with Squid. Insert a line similar to the one below to your CBQ config files placed in `/etc/sysconfig/cbq/`:

```
TIME=00:00-07:59;110Kbit/11Kbit
```

You can have multiple TIME parameters in your CBQ config files.

Be careful though, because there is a small bug in that `cbq.init-v0.6.2` script – it won't let you set certain times, for example 00:00–08:00! To make sure if everything is working correctly, start `cbq.init-v0.6.2`, and then within the time you set, type

/etc/rc.d/cbq.init−v0.6.2 timecheck

This is the example how the proper output should look like:

```
[root@mangoo rc.d]# ./cbq.init start; ./cbq.init timecheck **CBQ:  3:44:
class 10 on eth0 changed rate (20Kbit -> 110Kbit) **CBQ:  3:44: class 40
on ppp0 changed rate (15Kbit -> 110Kbit) **CBQ:  3:44: class 50 on eth0
changed rate (35Kbit -> 110Kbit)
```

In this example something went wrong, probably in the second config file placed in `/etc/sysconfig/cbq/`; second counting from the lowest number in its name:

```
[root@mangoo rc.d]# ./cbq.init start; ./cbq.init timecheck **CBQ:  3:54:
class 10 on eth0 changed rate (20Kbit -> 110Kbit) ./cbq.init: 08: value
too great for base (error token is "08")
```

The second way to make CBQ more intelligent is harder – it doesn't depend on time. You can read about it in the Linux 2.4 Advanced Routing HOWTO, and play with `tc` command.

6. Miscellaneous

6.1. Useful resources

Squid Web Proxy Cache

http://www.squid−cache.org

Squid 2.4 Stable 1 Configuration manual

http://www.visolve.com/squidman/Configuration%20Guide.html

http://www.visolve.com/squidman/Delaypool%20parameters.htm

Squid FAQ

http://www.squid−cache.org/Doc/FAQ/FAQ−19.html#ss19.8

cbq−init script

ftp://ftp.equinox.gu.net/pub/linux/cbq/

Linux 2.4 Advanced Routing HOWTO

http://www.linuxdoc.org/HOWTO/Adv−Routing−HOWTO.html

Traffic control (in Polish)

http://ceti.pl/~kravietz/cbq/

Securing and Optimizing Linux Red Hat Edition − A Hands on Guide

http://www.linuxdoc.org/guides.html

IPTraf

http://cebu.mozcom.com/riker/iptraf/

IPCHAINS

http://www.linuxdoc.org/HOWTO/IPCHAINS−HOWTO.html

Nylon socks proxy server

http://mesh.eecs.umich.edu/projects/nylon/

Indonesian translation of this HOWTO by Rahmat Rafiudin *mjl_id@yahoo.com*

http://raf.unisba.ac.id/resources/BandwidthLimitingHOWTO/index.html