

# How to manage bandwidth with OpenBSD PF queuing

August 12, 2018

## Abstract

Example for the configuration of an OpenBSD 5.9 based IPv4 network router with bandwidth management. The given example addresses a small home network with four users supporting multiple devices (e.g., notebooks, mobiles, tablets or desktops) per user. The Internet connection is implemented based on PPPoE dial-up line (e.g., ADSL).

## Introduction

### Disclaimer

The only one responsible for what you are doing is yourself. Simply following this how-to will not exclude you from this responsibility. In other words: do not blame me, if things go wrong.

### Objective

This is not a beginners guide to OpenBSD, PF, networking, the Internet, or computers in common. It addresses IT professionals.

### Versions

New versions of this paper might be found on [www.benjaminheckmann.de/howto/](http://www.benjaminheckmann.de/howto/).

# 1 OpenBSD Installation

## Hardware Preparation

As minimal setup for a router you will need:

- Two Network Cards (one for the connection to the Internet, the other for your local area network)
- One Hard Disk Drive with at least 5 GB capacity (recommended, to be able to update your installation)
- CD-ROM drive
- The other stuff (like a processor, memory, etc.; for further details see [www.openbsd.org/faq/faq1.html#Platforms](http://www.openbsd.org/faq/faq1.html#Platforms))

Download the files *install59.iso*, *src.tar.gz*, and *sys.tar.gz*. Create a bootable CD using the *install59.iso* file. For more details see [www.openbsd.org/faq/faq4.html#MkCD-ROM](http://www.openbsd.org/faq/faq4.html#MkCD-ROM).

## Basic Operating System Setup

Boot your system from the prepared bootable CD. Follow the install dialog:

1. (I)nstall
2. Keyboard layout: de
3. Hostname: router
4. Network interfaces: <dev0>
5. IPv4: 192.168.0.1
6. Netmask: 255.255.255.0
7. IPv6: none
8. <dev1> (unconfigured)
9. Default IPv4 route: none
10. DNS domain name: home.local
11. DNS nameservers: 127.0.0.1
12. Enter your password for the root account
13. Start sshd: yes
14. X Windows System: no

15. Change default console: no
16. Setup a user: no
17. Allow root ssh login: yes
18. root disk: wd0
19. Use whole disk: whole
20. Use (A)uto layout
21. Location of sets: cd0
22. Pathname: 5.9/amd64
23. Select sets:
  - bsd
  - bsd.rd
  - base59.tgz
  - comp59.tgz
  - man59.tgz
  - -game59.tgz
  - -xbase59.tgz
  - -xshare59.tgz
  - -xfont59.tgz
  - -xserv59.tgz
  - done
24. No SHA256.sig: yes
25. Location of additional sets: done
26. Timezone: Europe/Berlin
27. reboot

See [www.openbsd.org/faq/faq4.html](http://www.openbsd.org/faq/faq4.html), if you need more advice installing OpenBSD.

## Prepare OpenBSD updates

After booting your new OpenBSD system, log in, mount your CD-ROM and extract the OpenBSD sources for future updates in `/usr/src`. This will consume at least 2 GB disk space, so be prepared.

```
mount_cd9660 /dev/cd0c /mnt
cd /usr/src
tar xvzf /mnt/src.tar.gz
tar xvzf /mnt/sys.tar.gz
```

## Configuration of Basic OS Services

To enable routing, activate IP forwarding in the kernel. Also, configure the routing of GRE packages for VPNs using PPTP.

```
vi /etc/sysctl.conf
<...>
net.inet.gre.allow=1
net.inet.ip.forwarding=1
<...>
```

Configure the start of the firewall, nameserver, timeserver and ftp proxy.

```
vi /etc/rc.conf.local

named_flags=""
ntpd_flags=""
dhcpcd_flags=""
ftpproxy_flags=""
sendmail_flags=NO
inetd=NO
check_quotas=NO
```

Deactivate sendmail's cronjob.

```
crontab -e
<...>
#*/30 * * * * /usr/sbin/sendmail <...>
<...>
```

The automatic startup of the relevant services is now configured. The next step is the configuration of the specific service behavior.

```
vi /var/named/named.boot

options forward-only forwarders <ip0> <ip1> <...>
```

```
vi /etc/resolv.conf

lookup file bind
nameserver 127.0.0.1
```

```
vi /etc/ntp.conf

servers pool.ntp.org
```

```
rdate -ncv pool.ntp.org
```

In former how-tos the configuration of a DHCP server was detailed. This how-to obtains this feature.

For the configuration of the PPPoE dial-up, the according authentication data must be known.

```
vi /etc/hostname.pppoe0
```

```
inet 0.0.0.0 255.255.255.255 0.0.0.1 pppoe0 <dev2>  
authproto pap authname <user> authkey <password> up  
!/sbin/route add default 0.0.0.1
```

```
vi /etc/hostname.<dev1>
```

```
up
```

## 2 Configuration of Firewall and Bandwidth Management

The following PF configuration is an example for a small home network with four user. The ruleset supports multiple devices per user.

Former how-tos featured fixed port ranges for incoming traffic for specific clients. This feature is now deprecated.

```
mv /etc/pf.conf /etc/pf_ example.conf  
vi /etc/pf.conf
```

```
##  
## Interfaces  
##  
  
if_ext="pppoe0"  
if_int="<dev0>"  
  
##  
## Bandwidth  
##  
  
# Example: 160Kb upstream, 864Kb downstream (!  
bits, not bytes !)  
  
# Total upstream
```

```

bnd_up_max="160Kb"
# NAS, Router, etc. := 1/6
bnd_up_default="26Kb"
# User One := 1/4
bnd_up_usr1="40Kb"
# User Two := 1/4
bnd_up_usr2="40Kb"
# User Three := 1/6
bnd_up_usr3="27Kb"
# User Four := 1/6
bnd_up_usr4="27Kb"

# Total downstream
bnd_dn_max="864Kb"
# NAS, Router, etc. := 1/6
bnd_dn_default="144Kb"
# User One := 1/4
bnd_dn_usr1="216Kb"
# User Two := 1/4
bnd_dn_usr2="216Kb"
# User Three := 1/6
bnd_dn_usr3="144Kb"
# User Four := 1/6
bnd_dn_usr4="144Kb"

##
## Privileged clients
##

# User One
table <usr1_direct> { 192.168.2.21, 192.168.2.41, 192.168.2.50
}

# User Two
table <usr2_direct> { 192.168.2.22, 192.168.2.42 }
# User Three
table <usr3_direct> { 192.168.2.12, 192.168.2.32 }
# User Four
table <usr4_direct> { 192.168.2.11, 192.168.2.31 }

##
## Privileged networks
##

table <nets_priv> { 127.0.0.0/8, 192.168.0.0/16, 172.16.0.0/12,
10.0.0.0/8 }

```

```

##
## Default behavior
##

# Default response for block filters
set block-policy drop
# Bind states to interfaces
set state-policy if-bound
# Logging
set loginterface $if_ext
# Ignore traffic on the local interface
set skip on lo

##
## Upstream queues
##

# Parent queue
queue up_parent on $if_ext bandwidth $bnd_up_max
# NAS, Router, etc.
queue up_default parent up_parent bandwidth $bnd_up_default
default
# User One
queue up_usr1 parent up_parent bandwidth $bnd_up_usr1
# User Two
queue up_usr2 parent up_parent bandwidth $bnd_up_usr2
# User Three
queue up_usr3 parent up_parent bandwidth $bnd_up_usr3
# User Four
queue up_usr4 parent up_parent bandwidth $bnd_up_usr4

##
## Downstream queues
##

# Parent queue
queue dn_parent on $if_int bandwidth $bnd_dn_max
# NAS, Router, etc.
queue dn_default parent dn_parent bandwidth $bnd_dn_default
default
# User One
queue dn_usr1 parent dn_parent bandwidth $bnd_dn_usr1
# User Two
queue dn_usr2 parent dn_parent bandwidth $bnd_dn_usr2
# User Three
queue dn_usr3 parent dn_parent bandwidth $bnd_dn_usr3

```

```

# User Four
queue dn_usr4 parent dn_parent bandwidth $bnd_dn_usr4

##
## NAT
##

match out on $if_ext from $if_int:network to any nat-
to ($if_ext)

##
## Normalize traffic
##

# MTU adaption for VSDL via max-mss
match on $if_ext scrub (random-id max-mss 1452)

##
## Default filter
##

# Note: last matching rule wins => first rule blocks
all
block all

##
## External interface filtering
##

# Deny incoming or outgoing privileged network ad-
dress sets
block in quick on $if_ext from <nets_priv> block out
quick on $if_ext to <nets_priv>
# Allow incoming ping request to router and keep state
pass in quick on $if_ext inet proto icmp to ($if_ext)
icmp-type echoreq
# Allow outbound traffic, sort into queues and keep
state
pass out on $if_ext inet queue up_default
pass out on $if_ext inet from <usr1_direct> queue
up_usr1
pass out on $if_ext inet from <usr2_direct> queue
up_usr2
pass out on $if_ext inet from <usr3_direct> queue
up_usr3

```

```

    pass out on $if_ext inet from <usr4_direct> queue
up_usr4

##
## Internal interface filtering
##

# Redirect FTP client traffic
pass in quick on $if_int inet proto tcp to port 21 divert-
to 127.0.0.1 port 8021

# Allow outbound traffic, but do not track its state
pass in on $if_int inet no state

# Sort inbound traffic into queues
pass out on $if_int inet queue dn_default no state
pass out on $if_int inet to <usr1_direct> queue dn_usr1
no state
pass out on $if_int inet to <usr2_direct> queue dn_usr2
no state
pass out on $if_int inet to <usr3_direct> queue dn_usr3
no state
pass out on $if_int inet to <usr4_direct> queue dn_usr4
no state

##
## FTP proxy anchor
##

anchor "ftp-proxy/*"

##
## Deny spoofing
##

antispoof quick for { lo, $if_int }

```

### 3 Prepare OpenBSD Updates

The following scripts enable the update of the system in three steps. The *tmux* command can be used to prevent damage to the system in case of connection aborts on ssh sessions, while executing those scripts.

```
vi update-00_init.sh
```

```
#!/bin/csh
cd /usr
setenv CVS_CLIENT_PORT -1
setenv CVSROOT anoncvs@anoncvs.openbsd.org:/cvs
cvs -qd $CVSROOT get -rOPENBSD_5_9 -P src
```

```
vi update-01_sync.sh
```

```
#!/bin/csh
cd /usr/src
setenv CVS_CLIENT_PORT -1
cvs -q up -rOPENBSD_5_9 -Pd
```

```
vi update-02_kernel.sh
```

```
#!/bin/csh
cd /usr/src/sys/arch/amd64/conf
/usr/sbin/config GENERIC
cd /usr/src/sys/arch/amd64/compile/GENERIC
make clean && make
cd /usr/src/sys/arch/amd64/compile/GENERIC
make install
```

```
vi update-03_binaries.sh
```

```
#!/bin/csh
rm -rf /usr/obj/*
cd /usr/src
make obj
cd /usr/src/etc && env DESTDIR=/ make distrib-
dirs
cd /usr/src
make build
```

```
chmod ug+x update-*.sh
```

## 4 Verifying queue effectivity

Some useful commands to monitor the usage of the configured queues:

- `pfctl -vvs queue`
- `systat queues`
- `pfctl -vvs rules`

The first two commands enable the monitoring of the queue usage. The last command may help to sort out, why queuing might not work in your ruleset.