

# FreeBSD PF firewall using gigabit Ethernet

Building a firewall using cheap Amd64 and Intel Gigabit Ethernet!

The purpose of this page is

- to document a real life experiment done using FreeBSD and Gigabit equipment

I did this experiment while building and finishing a firewall for DKUUG, the Danish Unix User Group.

## Equipment used

- IBM Thinkpad model X31 named Timon
- Apple Powerbook G4 1,25GHz named Bigfoot
- Amd64 based PC with 3 Intel Gigabit Ethernet, only two used named Sobek
- Asus GigaX 1108 8-port Gigabit switch
- Cat 5E 1 meter cables

## Software used

- OpenBSD 3.9-beta
- FreeBSD 6.0 with PF filtering
- Mac OS X 10.4
- NetStrain 3.0 (c) 2002 Christoph Pfisterer

## Method used - not very scientific, but fun nonetheless :-)

1. connect two laptops directly with 1 meter cables - measure maximum throughput
2. connect two laptops with a FreeBSD PF firewall, firewall disabled - maximum throughput routing

## FreeBSD firewall using gigabit ethernet

3. connect two laptops with a FreeBSD PF firewall, firewall enabled - maximum throughput filtering firewall

## Measurements done

All measurements are done using Netstrain which is a nice and simple tool - some may find it too simple.

When you use netstrain you start a netstraind daemon at one end and then run the netstrain client at the other. The client is then run in either send, recv or both mode.

Sample run from bigfoot with netstraind started at Timon on IP address 10.3.4.100:

```
bigfoot# netstrain -4 10.3.4.100 12345 send
NetStrain 3.0 (c) 2002 Christoph Pfisterer
Looking up hostname 10.3.4.100...
Connecting to 10.3.4.100 port 12345 using IPv4...
Connected
sent: 540M, 28332.1K/s total, 29553.7K/s current
recv'd: 0B, 0B/s total, 0B/s current
^C
```

Since this method is very dynamic - you can see the numbers being updated I needed to find a way to write down results using a fixed set of rules to get results that I could trust.

The rules I used were:

- netstraind is run at Timon as root with port 12345 listening
- netstrain is run at Bigfoot as root
- run 3 times for each send, recv and both settings and each time for at least 15-20 seconds
- note down the result shown when doing a control-c and breaking the run

## Results

Several scenarios were tested from the most simple connection to the real life firewall setup - with only one server and one client.

The documented setups are:

1. Two laptops Bigfoot and Timon directly connected with 1 meter Cat 5E cable
2. Two laptops Bigfoot and Timon connected with 1 meter Cat 5E cables to Asus switch port 1 and 8
3. Two laptops Bigfoot and Timon connected with cables/switch and Sobek Firewall

## Results

These results presented are from above experiments with the three setups:  
All results are K/s as reported by Netstrain program when ctrl-c breaking the test after at least 15-20 seconds of runtime.

## Production use

Now the firewall is put into production on a connection at ISP TDC and the results I get from another administrator is about 7.9 MB/s from a FTP server behind the firewall to a server on the core of the ISP Cybercity. This is much better than the old firewall performance so we are pretty happy.  
BTW we use ftpsesame for this firewall from the ports collection on FreeBSD - works nicely with our servers.

## Detailed information

- [dmesg](#) from Timon
- [ifconfig](#) from Timon
- [dmesg](#) from Bigfoot
- [ifconfig](#) from Bigfoot
- [dmesg](#) from Sobek
- [ruleset](#) used in 3b)

## Notes

1. I seem to remember getting something like 60-70.000 at most when doing a similar experiment from Bigfoot to Fluffy which is another Amd64 based server
2. I tested all the cables in the Bigfoot-Timon direct connect experiment, to make sure they were all in working condition
3. I saw very stable numbers in the first experiments and more irregular numbers when doing the firewall tests
4. I would have liked to do the experiment with IPv6 but had various problems that prevented that :-)
5. I would have liked to do the experiments using jumbo-frames of 9000 bytes, but didn't find out how to enable it on my Mac OS X :-)

There are multiple ways to redo this experiment and test jumboframes and IPv6 - I will do that when I find the time.