

OpenBSD network performance boosting

At the OpenBSD hackathon C2K7 the network performance of OpenBSD was boosted tremendously. Read more about this work in the presentation: The presentation slides (Faster packets: 10GE, faster pf and other fun stuff) are up at <http://www.openbsd.org/papers/cuug2007/>.

The purpose of this page is

- to show that indeed the performance changed, verify claims :-)
- to measure just how much I can put through a Soekris 4801 with OpenBSD

Initial testing

Using some systems with pretty new kernels I did some initial testing. The kernels are all from beginning of May. I thought I would make new kernels, update my systems and then leave them until next release.

```
$ dmesg | head OpenBSD 4.1-current (RAID.MP) #0: Wed May 9 19:10:35 CEST 2007 root@sylvester.kramse.dk:/sys/arch/amd64/compile/RAID.MP
```

Systems used:

- Timon Thinkpad X31 with em0 Gbit running OpenBSD/i386
- Fluffy Asus K8V-Deluxe motherboard with AMD Athlon(tm) 64 Processor 3200+ ("AuthenticAMD" 686-class, 1024KB L2 cache) 2.01 GHz running OpenBSD/Amd64
- Fluffy Asus M2N-SLI/Deluxe with dual nfe Gbit running OpenBSD/Amd64
- Sylvester Shuttle Amd64 system running OpenBSD/amd64 - buildhost for Amd64

Iperf initial run

I tried running Iperf with Timon as server and recieved connections from three hosts

Most testing was done across a small Asus GigaX 1108 8-port Gigabit switch.

- 10.0.42.23 Fluffy - connected via Linksys SRW-2016 "the house core switch"
- 10.0.42.34 Sylvester - connected via Apple Airport Express and Airport

OpenBSD network performance boosting

Extreme using WDS

- 10.0.42.41 Flaffy - directly connected to Asus GigaX switch - this is the main test

```
hlk@timon hlk$ iperf -s
```

```
-----  
Server listening on TCP port 5001  
TCP window size: 16.0 KByte (default)  
-----  
[ 4] local 10.0.42.42 port 5001 connected with 10.0.42.23 port 1969  
[ 4] 0.0- 8.1 sec 88.5 MBytes 91.9 Mbits/sec  
[ 4] local 10.0.42.42 port 5001 connected with 10.0.42.23 port 25811  
[ 4] 0.0- 2.0 sec 22.1 MBytes 91.6 Mbits/sec  
[ 4] local 10.0.42.42 port 5001 connected with 10.0.42.23 port 42345  
[ 4] 0.0- 4.7 sec 50.8 MBytes 91.6 Mbits/sec  
[ 4] local 10.0.42.42 port 5001 connected with 10.0.42.34 port 46168  
[ 4] 0.0- 9.6 sec 8.65 MBytes 7.56 Mbits/sec  
[ 4] local 10.0.42.42 port 5001 connected with 10.0.42.34 port 25480  
[ 4] 0.0-10.1 sec 8.66 MBytes 7.22 Mbits/sec  
[ 4] local 10.0.42.42 port 5001 connected with 10.0.42.34 port 15594  
[ 4] 0.0-10.1 sec 8.59 MBytes 7.16 Mbits/sec  
[ 4] local 10.0.42.42 port 5001 connected with 10.0.42.41 port 10762  
[ 4] 0.0-10.1 sec 475 MBytes 397 Mbits/sec  
[ 4] local 10.0.42.42 port 5001 connected with 10.0.42.41 port 44447  
[ 4] 0.0-10.1 sec 486 MBytes 405 Mbits/sec  
[ 4] local 10.0.42.42 port 5001 connected with 10.0.42.41 port 6501  
[ 4] 0.0-10.1 sec 483 MBytes 403 Mbits/sec
```

Pretty OK I think.

Update the kernels on Timon and Flaffy

Since I now had numbers from Timon and Flaffy I installed the fresh newly compiled kernels (cvsync today June 4th).

First run after updating only the kernel on Timon:

```
hlk@timon hlk$ iperf -s
```

```
-----  
Server listening on TCP port 5001  
TCP window size: 16.0 KByte (default)  
-----  
[ 4] local 10.0.42.42 port 5001 connected with 10.0.42.23 port 22594  
[ 4] 0.0-10.0 sec 109 MBytes 91.3 Mbits/sec  
[ 4] local 10.0.42.42 port 5001 connected with 10.0.42.23 port 15021  
[ 4] 0.0-10.0 sec 109 MBytes 91.6 Mbits/sec  
[ 4] local 10.0.42.42 port 5001 connected with 10.0.42.41 port 1812  
[ 4] 0.0-10.1 sec 601 MBytes 501 Mbits/sec  
[ 4] local 10.0.42.42 port 5001 connected with 10.0.42.41 port 48626  
[ 4] 0.0-10.1 sec 601 MBytes 501 Mbits/sec
```

As you can see the performance went from around 400 Mbits/sec to about 500 Mbit/sec! Updating Flaffy with a new kernel really didn't do much - I guess the CPU was fast enough to begin with.

Inserting a bridging Soekris 4801

I inserted a Soekris 4801-50 between Timon and the switch configured without firewall and bridging. The kernel was initially from April 16th 2007 - so I expected normal performance. This is the performance I got:

```
hlk@timon hlk$ iperf -s
-----
Server listening on TCP port 5001
TCP window size: 16.0 KByte (default)
-----
[ 4] local 10.0.42.42 port 5001 connected with 10.0.42.41 port 14206
[ 4] 0.0-10.0 sec 59.8 MBytes 50.1 Mbits/sec
[ 4] local 10.0.42.42 port 5001 connected with 10.0.42.41 port 34729
[ 4] 0.0-10.0 sec 59.5 MBytes 49.9 Mbits/sec
[ 4] local 10.0.42.42 port 5001 connected with 10.0.42.41 port 30486
[ 4] 0.0-10.0 sec 59.4 MBytes 49.8 Mbits/sec
```

Not really bad, as I think we normally say a Soekris can do about 40 Mbit/sec.

Boosting the bridge

Just changing the kernel to the new one - same as Timon and redoing the testing:

```
hlk@timon hlk$ iperf -s
-----
Server listening on TCP port 5001
TCP window size: 16.0 KByte (default)
-----
[ 4] local 10.0.42.42 port 5001 connected with 10.0.42.41 port 41407
[ 4] 0.0-10.0 sec 51.6 MBytes 43.2 Mbits/sec
[ 4] local 10.0.42.42 port 5001 connected with 10.0.42.41 port 16704
[ 4] 0.0-10.0 sec 51.5 MBytes 43.2 Mbits/sec
[ 4] local 10.0.42.42 port 5001 connected with 10.0.42.41 port 42357
[ 4] 0.0-10.0 sec 51.4 MBytes 43.1 Mbits/sec
```

WTF! The performance went down?! hmmm need further testing!

Further testing

I need to try adding the firewall and also test routing performance. I published this ASAP, as some people had questions about Soekris performance at the BSD-DK IRC channel, hi guys :-)

Notes

While not recommended for all I do build my own kernels, since I need a buildserver for RAIDframe, JDK and other things anyway. The kernels used in the above experiments are either built from GENERIC or RAID.MP which is my GENERIC+RAIDframe. No exceptional tuning was done! YMMV but some further tuning might be available.

The network used was not 100% clean from other traffic, the Asus GigaX switch was used for netradio, e-mailing and surfing during the experiments.