Welcome to

Open Source projects with 10Gbit

Network Camp 2012

Henrik Lund Kramshøj hlk@kramse.org

THECAMP.DK - 7 open source days

Plan for foredraget

30 min Network introduction Kristen Nielsen
- shared view of networking, common terms

1,5t The Network Camp 2012 experiments

Networking hardware

Operating systems, the network stacks and TCP/IP problems

Network measurements: performance testing with iperf, tcpbench m.fl.

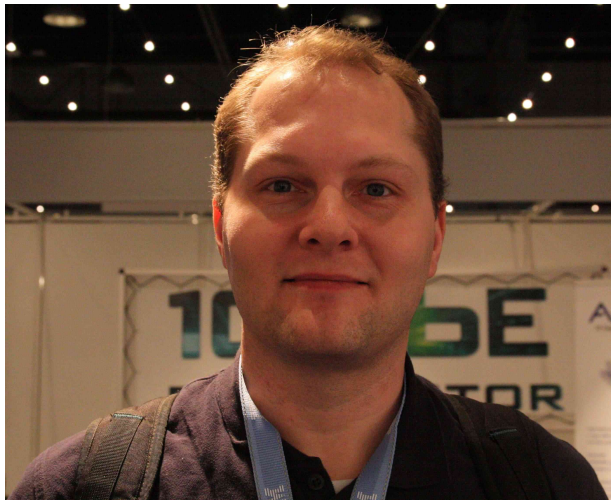
More application testing and hacker tools

My current results for Linux, FreeBSD and OpenBSD

Troubleshooting

Conclusion

Contact information



- Henrik Lund Kramshøj, internet samurai
- Email: hlk@solido.net Mobile: +45 2026 6000 IRC: hlkv6
- Cand.scient the Computer Science Department at the University of Copenhagen, DIKU
- 2003 - 2010 Independent security consultant
- 2010 - owner and partner in Solido Networks ApS

Background for this presentation

Lab testing 10Gbit servers with various open source operating systems

Can they support 10Gbps?

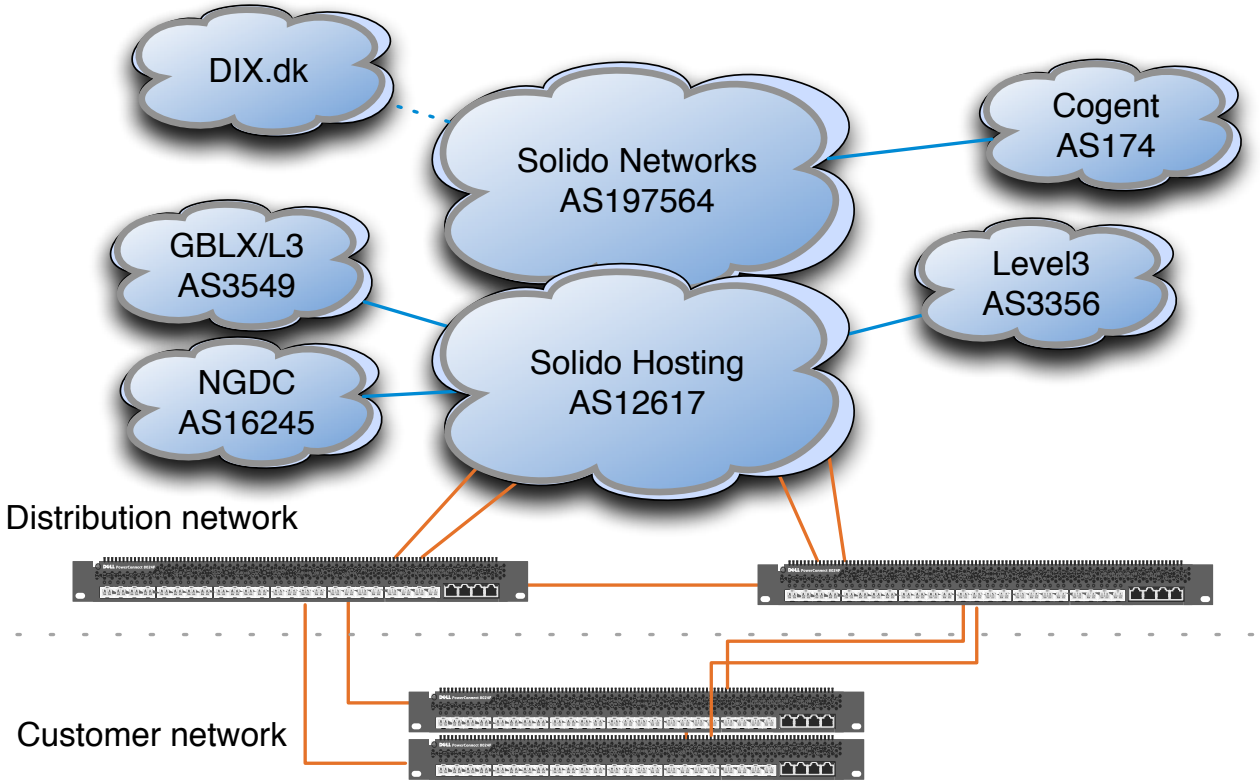
What is needed to reach 10Gbit?

Goal - be able to configure high performance servers with octo-core and 100Gb memory for 100.000s of connections or requests/sec supporting DNS, HTTP and email

On commodity hardware, like regular blade server

Some results from this is going directly in production after The Camp 2012
Going to roll out across +500 servers in Hosting
- updating the existing tuning settings

My networks



I co-admin AS12617 and AS59469

`http://bgp.he.net/AS12617` `http://bgp.he.net/AS59469`

My environments: europe

about 26ms latency - measured using ping

Networking hardware

I bought and configured the following systems based on Shuttle:

- Shuttle SX58H7 Pro Barebone
- Chipset Intel X58 + ICH10R
- Memory 4 x 240pin DDR3 DIMM slots,4GB per DIMM(Max 16GB)
  Support Triple channel DDR3 1066/1333/1600(OC)MHz
- Expansion slot (2) PCI-E X16 slot
- Ethernet (2) Realtek 8111E 10Mb/s,100Mb/s,1Gb/s operation

Actual configuration

- Shuttle SX58H7 Pro Barebone
- CPU Intel Core i7 - 960 (3.2GHz) (4core) (8MB) (130W) (LGA1366) (45nm)
- 4GB, DDR3, DIMM 240-pin, 1333 MHz (PC3-10600), CL9
- Club 3D Nvidia GeForce GT520 1GB PCI-E - SMALL - one slot only
- Dual port Intel 10GbE SFP+ (82599 chip)

This is just a sample system, let me know if you find better portable systems

Operating systems

My primary interests are:

- Linux - used for web hosting, DNS
- FreeBSD - used for Varnish and stuff
- OpenBSD - which we use mostly for DNS, NTP and *other infrastructure servers*

I will have to redo these experiments with Windows 2008R8 and Windows 7 clients

Note: This presentation is based on dual-port Intel 82599 NICs
Later this year I will probably be testing various other NICs, like Myricom, Broadcom, Brocade and maybe even Napatech

This is not a which OS is fastest, but more focus on how to achieve good network performance

IP network stacks

OSI Reference Model

Application
Presentation
Session
Transport
Network
Link
Physical

Internet protocol suite

Applications HTTP, SMTP, FTP, SNMP,	NFS
	XDR
	RPC
TCP UDP	
IPv4 IPv6 ICMPv6 ICMP	
ARP RARP MAC	
Ethernet token-ring ATM ...	

TCP/IP features

IP/TCP is old RFC-793 September 1981

Implemented on the internet circa 1983

TCP over low speed data links to High speed 10Gbit Ethernets

Soon we will have 100Gbps links in production

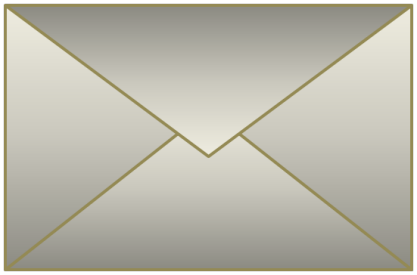
I have switches on camp with 40Gbit ports!

Many keywords on the internet today: Ethernet, Layer2, Layer3, SNMP, OSPF, BGP, firewalls, routers, switches, syslog, ACL, security, denial of service, syn flooding, back-track, Nmap, portscan, Metasploit, 802.1q, 802.11, 802.1x, STP, BPDU guard, IPv4, IPv6, ICMPv6, NTP, Cisco IOS, 802.3ad

I will not repeat a lot of slides about protocols, use Wikipedia has great pages about most network protocols

The network characteristic

then …. and now



Networks have evolved from mail being the primary application

today we expect to stream high definition content

Hosts on the internet maybe 800 million mobile-only in 2015?

Unified communications - *everything over networks*
`http://en.wikipedia.org/wiki/Unified_communications`

Problems - incomplete list!

Too much bandwidth Long fat networks (LFNs, pronounced ”elefan(t)s”)
Even reaching 1Gbit transfer over TCP between Copenhagen and Luxembourg with 25ms latency can be tough

TCP Slow start - to little data sent initially
`http://en.wikipedia.org/wiki/Slow-start`

Buffer bloat - efficient buffering, break TCP’s congestion-avoidance algorithms
`http://en.wikipedia.org/wiki/Bufferbloat`

Do we have tools for solving network stack problems

Today we have:

- Huge memory available +16Gb in single user work stations, more in servers
  We can use larger buffers
- RFC1323 TCP Extensions for High Performance - May 1992 ☺
  includes Negotiating TCP Large Windows options (RFC1323)
- RFC2018 TCP Selective Acknowledgments (SACK)
- RFC3390 Increasing TCP's Initial Window
- TCP sliding window - INCREASE the buffer captain
- Path MTU Discovery - Go big or Go home
  minimum IP datagram size that a host must be able to receive IPv4 576, IPv6 1280

Conclusion: do not expect things to work out of the box in every case - test

Bandwidth-delay product TCP

Examples

- Moderate speed satellite network: 512 kbit/s, 900 ms RTT
  $B \times D = 512 \times 10^3$ b/s $\times$ 900×10^{-3} s = 460,800 b.
- Residential DSL: 2 Mbit/s, 50 ms RTT
  $B \times D = 2 \times 10^6$ b/s $\times$ 50×10^{-3} s = 100×10^3 b, or 100 kb, or 12.5 kB.
- Mobile broadband (HSDPA): 6 Mbit/s, 100 ms RTT
  $B \times D = 6 \times 10^6$ b/s $\times$ 10^{-1} s = 6×10^5 b, or 600 kb, or 75 kB.
- Residential ADSL2+: 20 Mbit/s (from DSLAM to residential modem), 50 ms RTT
  $B \times D = 20 \times 10^6$ b/s $\times$ 50×10^{-3} s = 10^6 b, or 1 Mb, or 125 kB.
- High-speed terrestrial network: 1 Gbit/s, 1 ms RTT
  $B \times D = 10^9$ b/s $\times$ 10^{-3} s = 10^6 b, or 1 Mb, or 125 kB.

http://en.wikipedia.org/wiki/Bandwidth-delay_product

Network measurements

The need for testing - crucial for success!

Well known tool iperf

Others exist like: Netperf, tcpbench

FTP, SCP, SFTP - Warning: when disk/storage I/O is introduced less accurate network results!
Still this is the next step for the server people ;-) (and they will know the network works, and must tune database exports, disk I/O, iSCSI, SAN etc.)

Smokeping - latency monitoring

Why use popular tools - other people will recognize results immediately

Trust nobody, use the switch port for measurements


```
force10-2#sh interfaces Fo 0/60 | grep Mbits
     Input 9869.00 Mbits/sec,     813004 packets/sec, 25.00% of line-rate
     Output 35.00 Mbits/sec,      62589 packets/sec, 0.11% of line-rate
force10-2#sh interfaces Tengigabitethernet 0/0 | grep Mbits
     Input 35.00 Mbits/sec,      62616 packets/sec, 0.45% of line-rate
     Output 9869.00 Mbits/sec,     812987 packets/sec, 99.90% of line-rate
```

Were do you measure?

What about different tools - some report Mbits per second, other Mbytes

I will use the switch port for precise measurements, if needed

Output shown is from Force10 S4810 switch

iperf

iperf was developed by NLANR/DAST as a modern alternative for measuring maximum TCP and UDP bandwidth performance.

Client server based

Both TCP and UDP, and allows various tuning parameters

Make sure to use the latest version - iperf-2.0.5.tar.gz

Precompiled executable for Windows are available at various places, ask me :-)

Used by all danish ISPs, and the rest of the world

Information: `http://en.wikipedia.org/wiki/iperf`
Source at: `http://sourceforge.net/projects/iperf/`

iperf server


```
[hlk@patchman ~]$ sudo iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 109.238.48.53 port 5001 connected with 109.238.49.65 port 50395
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.0 sec  1.08 GBytes   931 Mbits/sec
```

iperf client


```
solido@solido01:~$ sudo iperf -c 109.238.48.53
------------------------------------------------------------
Client connecting to 109.238.48.53, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 109.238.49.65 port 50395 connected with 109.238.48.53 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.08 GBytes    932 Mbits/sec
```

Recommendations:

start the server, keep it running

run for at least 60 seconds

typical use `iperf -t 60 -i 5 -c 10.0.10.30`

Tuning the network

When you have used iperf or similar to perform baseline testing

start tuning your engines ☺

The main tuning options will be kernel related and today done via `/etc/sysctl.conf` from 4.4BSD now also on Linux, see `http://en.wikipedia.org/wiki/Sysctl`

Other tuning options:

- Kernel upgrade - new kernels may be better - or worse!
- Driver upgrade - download newest driver from Intel, install and test, may be better - or worse!
- Driver options - intel driver has some parameters, some compile-time others dynamic /boot/loader.conf used on FreeBSD to load ixgbe driver with options at boot time

Disabling unnecessary daemons and features

Testing methodology

When testing:

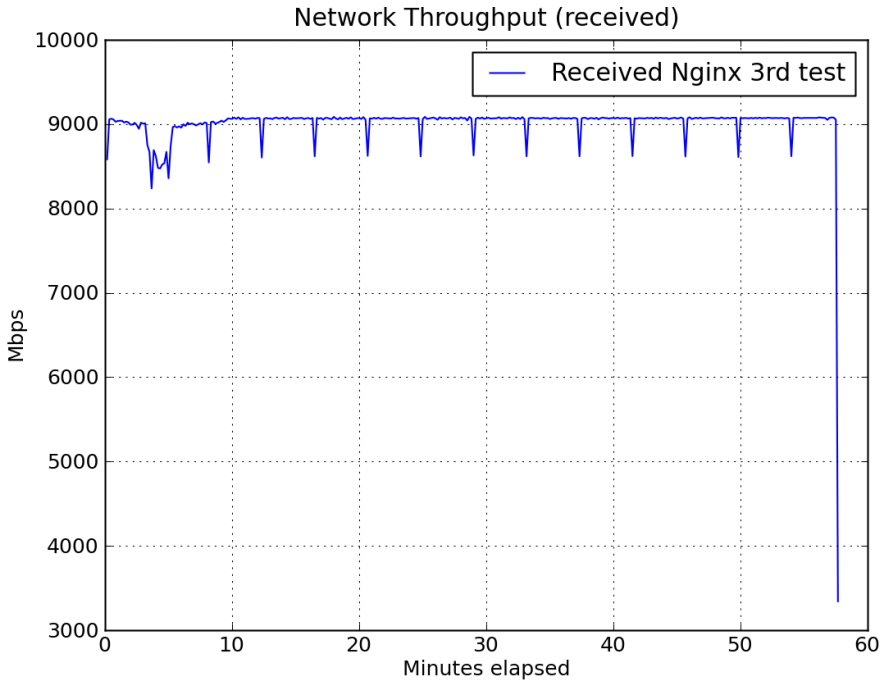
- Do a baseline before optimizing
- Do initial testing with the usual stuff - /etc/sysctl.conf
- Research, google, mailinglists for operating system - have others succeeded
- Take small steps, confirm by turning on and off
- Try switching your testing tool - the tool might be bad for this combination of OS, NIC, hardware
- Monitor the operating system, especially interrupts, but also logs, console etc.

Rinse repeat - redo tests often, redo the baseline test again

Trying to formalize what we do, but most importantly

keep notes, what you did, what was changed

More application testing



Tsung can be used to stress HTTP, WebDAV, SOAP, PostgreSQL, MySQL, LDAP and Jabber/XMPP servers `http://tsung.erlang-projects.org/`

Apache benchmark included with Apache

Hacker tools

All public web sites will receive bad traffic

You need to run hacker tools to simulate attacks, and to know how your environment react to bad traffic

Install Backtrack Linux `http://www.backtrack-linux.org/`

Run hacker tools from Backtrack, install those which are not included

More inspiration can be found in my other presentations:
`http://files.kramse.org/tmp/creative-packets-screen.pdf`
`http://www.version2.dk/blogs/henrik-lund-kramshoej`

Results from Linux after tuning


```
hlk@xpc03:~$ sudo iperf -t 60 -i 5 -c 10.0.10.20
------------------------------------------------------------
Client connecting to 10.0.10.20, TCP port 5001
TCP window size: 9.54 MByte (default)
------------------------------------------------------------
[  3] local 10.0.10.30 port 49131 connected with 10.0.10.20 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 5.0 sec  5.52 GBytes  9.49 Gbits/sec
[  3]  5.0-10.0 sec  5.53 GBytes  9.50 Gbits/sec
[  3] 10.0-15.0 sec  5.52 GBytes  9.49 Gbits/sec
[  3] 15.0-20.0 sec  5.53 GBytes  9.49 Gbits/sec
[  3] 20.0-25.0 sec  5.53 GBytes  9.49 Gbits/sec
[  3] 25.0-30.0 sec  5.52 GBytes  9.49 Gbits/sec
[  3] 30.0-35.0 sec  5.52 GBytes  9.49 Gbits/sec
[  3] 35.0-40.0 sec  5.53 GBytes  9.49 Gbits/sec
[  3] 40.0-45.0 sec  5.53 GBytes  9.50 Gbits/sec
[  3] 45.0-50.0 sec  5.52 GBytes  9.49 Gbits/sec
[  3] 50.0-55.0 sec  5.53 GBytes  9.50 Gbits/sec
[  3]  0.0-60.0 sec  66.3 GBytes  9.49 Gbits/sec
```

Tested with BackTrack Linux distribution with this uname -a:

```
Linux xpc02 3.2.6 #1 SMP Fri Feb 17 10:34:20 EST 2012 x86_64 GNU/Linux
```

against Ubuntu Server with this uname -a:

```
Linux xpc03 3.2.0-26-generic #41-Ubuntu SMP Thu Jun 14 17:49:24 UTC 2012 x86_64
```

Tuning process performed on Linux

Baseline testing - no optimizations

Update to latest kernel/driver

Update /etc/sysctl.conf with some tweaks

Monitor and watch using ifpps, vmstat etc. for sign of bottle-necks

Ask friends - cry for help. Received good help from Brian Lagoni and others

ifpps


```
Kernel net/sys statistics for eth2, t=8.00s
  RX:             0.000 MiB/t           0 pkts/t          0 drops/t          0 errors/t
  TX:           658.223 MiB/t     9326989 pkts/t          0 drops/t          0 errors/t

  RX:             0.000 MiB             0 pkts            0 drops            0 errors
  TX:         11978.994 MiB     169741693 pkts            0 drops            0 errors

  SYS:             4342 cs/t         2.7% mem               4 running          0 iowait
  CPU0:            0.0% usr/t        0.1% sys/t        99.9% idl/t          0.0% iow/t
  CPU1:            0.2% usr/t        1.0% sys/t        98.8% idl/t          0.0% iow/t
  CPU2:            0.0% usr/t        0.0% sys/t       100.0% idl/t          0.0% iow/t
  CPU3:            0.0% usr/t        0.0% sys/t       100.0% idl/t          0.0% iow/t
  CPU4:            0.0% usr/t        2.7% sys/t        97.3% idl/t          0.0% iow/t
  CPU5:           17.4% usr/t       81.5% sys/t         1.1% idl/t          0.0% iow/t
  CPU6:            0.0% usr/t        0.0% sys/t       100.0% idl/t          0.0% iow/t
  CPU7:           15.0% usr/t       85.0% sys/t         0.0% idl/t          0.0% iow/t
```

ifpps while running test with tcpreplay - no responses sent back

awesome for showing bits, packets per sec, interrupts

http://netsniff-ng.org/

Current Linux configuration

/etc/sysctl.conf - much inspiration from Intel driver README:

```
# General 10 gigabit tuning:
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.core.optmem_max = 524287
net.core.rmem_default = 524287
net.core.wmem_default = 524287
net.ipv4.tcp_mem = 10000000 10000000 10000000
net.ipv4.tcp_rmem = 10000000 10000000 10000000
net.ipv4.tcp_wmem = 10000000 10000000 10000000
net.ipv4.tcp_syncookies = 1
# this gives the kernel more memory for tcp
# which you need with many (100k+) open socket connections
net.ipv4.tcp_mem = 50576   64768   98152
net.core.netdev_max_backlog = 50000
```

Tested using BackTrack and Ubuntu Server

Tested with BackTrack Linux distribution with this uname -a:

```
Linux xpc02 3.2.6 #1 SMP Fri Feb 17 10:34:20 EST 2012 x86_64 GNU/Linux
```

against Ubuntu Server with this uname -a:

```
Linux xpc03 3.2.0-26-generic #41-Ubuntu SMP Thu Jun 14 17:49:24 UTC 2012 x86_64
```

Results from XenServer (Linux) after tuning


```
[root@xen01 ~]# iperf -t 60 -i 5 -c 10.0.10.30
------------------------------------------------------------
Client connecting to 10.0.10.30, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 10.0.10.20 port 42256 connected with 10.0.10.30 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 5.0 sec  5.37 GBytes  9.22 Gbits/sec
[  3]  5.0-10.0 sec  5.39 GBytes  9.25 Gbits/sec
[  3] 10.0-15.0 sec  5.28 GBytes  9.07 Gbits/sec
[  3] 15.0-20.0 sec  5.37 GBytes  9.22 Gbits/sec
[  3] 20.0-25.0 sec  5.35 GBytes  9.19 Gbits/sec
[  3] 25.0-30.0 sec  5.38 GBytes  9.24 Gbits/sec
[  3] 30.0-35.0 sec  5.44 GBytes  9.34 Gbits/sec
[  3] 35.0-40.0 sec  5.40 GBytes  9.27 Gbits/sec
[  3] 40.0-45.0 sec  5.29 GBytes  9.10 Gbits/sec
[  3] 45.0-50.0 sec  5.23 GBytes  8.98 Gbits/sec
[  3] 50.0-55.0 sec  5.43 GBytes  9.34 Gbits/sec
[  3] 55.0-60.0 sec  5.32 GBytes  9.15 Gbits/sec
[  3]  0.0-60.0 sec  64.2 GBytes  9.20 Gbits/sec
```

Data from latest XenServer 6.2 with updates - update for Intel driver

Getting the last bits

Some extra sysctls were tuned:

```
kernel.shmmax = 4294967295
kernel.shmall = 268435456
fs.aio-max-nr = 444416
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_mem = 50576   64768   98152
net.core.netdev_max_backlog = 10000
```

Results from XenServer (Linux) after last tuning


```
[root@xen01 ~]# iperf -t 60 -i 5 -c 10.0.10.30
------------------------------------------------------------
Client connecting to 10.0.10.30, TCP port 5001
TCP window size: 64.0 KByte (default)
------------------------------------------------------------
[  3] local 10.0.10.20 port 32852 connected with 10.0.10.30 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 5.0 sec  5.44 GBytes  9.34 Gbits/sec
[  3]  5.0-10.0 sec  5.47 GBytes  9.40 Gbits/sec
[  3] 10.0-15.0 sec  5.47 GBytes  9.40 Gbits/sec
[  3] 15.0-20.0 sec  5.47 GBytes  9.40 Gbits/sec
[  3] 20.0-25.0 sec  5.48 GBytes  9.41 Gbits/sec
[  3] 25.0-30.0 sec  5.48 GBytes  9.41 Gbits/sec
[  3] 30.0-35.0 sec  5.46 GBytes  9.39 Gbits/sec
[  3] 35.0-40.0 sec  5.48 GBytes  9.41 Gbits/sec
[  3] 40.0-45.0 sec  5.47 GBytes  9.41 Gbits/sec
[  3] 45.0-50.0 sec  5.48 GBytes  9.42 Gbits/sec
[  3] 50.0-55.0 sec  5.47 GBytes  9.40 Gbits/sec
[  3] 55.0-60.0 sec  5.47 GBytes  9.40 Gbits/sec
[  3]  0.0-60.0 sec  65.7 GBytes  9.40 Gbits/sec
```

Squeeeeeezing the last bits from the XenServer :-)

Also verified a CentOS virtuel server could reach +9Gbits/sec!

Results from FreeBSD after tuning


```
root@xpc03:/root # iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 4.00 MByte (default)
------------------------------------------------------------
[  4] local 10.0.10.30 port 5001 connected with 10.0.10.20 port 53080
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-60.0 sec  42.9 GBytes  6.15 Gbits/sec
```

Note: the large TCP window size

Some room for improvement.

Results from FreeBSD after tuning


```
root@xpc02:/root # netperf -H 10.0.10.30 -t UDP_STREAM
MIGRATED UDP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET
to 10.0.10.30 (10.0.10.30) port 0 AF_INET
Socket  Message  Elapsed      Messages
Size    Size     Time         Okay Errors   Throughput
bytes   bytes    secs            #      #   10^6bits/sec

  9216    9216   10.00      1234419      0    9100.60
 42080           10.00      1234419           9100.60
```

So FreeBSD can go closer to wirespeed 10G when sending UDP - less work

Tuning process performed on FreeBSD

Baseline testing - no optimizations

Update to latest kernel/driver, from 9.0-RELEASE to 10-current

Update /etc/sysctl.conf with some tweaks
/etc/sysctl.conf: `kern.ipc.nmbclusters=262144`
/etc/sysctl.conf: `kern.ipc.nmbjumbop=262144`

Noticed warnings on console `hw.intr_storm_threshold=9000`

Update /boot/loader.conf with some tweaks

Remove WITNESS - kernel audit of semaphores and other stuff

Remove USB3 xhci driver - ate about 20-50.000 interrupts when idle!

Monitor and watch using systat, vmstat etc. for sign of bottle-necks

Ask friends - cry for help. Received good help from tykling, batmule and others

Current FreeBSD configuration

/etc/sysctl.conf

```
# Increase the network buffers
kern.ipc.nmbclusters=262144
kern.ipc.maxsockbuf=4194304
hw.intr_storm_threshold=9000
kern.ipc.nmbjumbop=262144
```

/boot/loader.conf

```
ixgbe_load="YES"
hw.ixgbe.txd=4096
hw.ixgbe.rxd=4096
```

We should go back and revisit FreeBSD 9.0-RELEASE for confirming which tuning option did most

Results from OpenBSD

Before tuning:

```
root@xpc03:root# tcpbench -n 4 -s
  elapsed_ms           bytes           mbps    bwidth
...
Conn:    4 Mbps:     2784.465 Peak Mbps:     2799.272 Avg Mbps:      696.116
         4000       86687512       693.500   25.00%
         3999       88704632       709.637   25.58%
         3999       88733536       709.868   25.59%
         3999       82607488       660.860   23.82%
Conn:    4 Mbps:     2773.865 Peak Mbps:    2799.272 Avg Mbps:      693.466
```

Results from OpenBSD after basic sysctl tuning


```
Output from tcpbench was similar to:
  elapsed_ms          bytes          mbps    bwidth
...
     353030       97605896       780.847   25.01%
     353030       97517512       780.140   24.99%
     353030       97517568       780.921   24.99%
     353029       97600864       781.589   25.01%
Conn:    4 Mbps:     3123.497 Peak Mbps:   3159.770 Avg Mbps:      780.874
```

Results from OpenBSD after intel card tuning

Increased hardware queues
TxDescriptors/RxDescriptors from 256 to 512 in `src/sys/dev/pci/if_ix.h`

```
root@xpc02:root# tcpbench -n 4 10.0.10.30
 elapsed_ms           bytes           mbps   bwidth
...
Conn:    4 Mbps:     3296.798 Peak Mbps:     3341.420 Avg Mbps:      824.200
       4006      107893344      863.147   26.49%
       4006       95005528      760.044   23.33%
       4006      110895272      887.162   27.23%
       4005       93440408      747.523   22.95%
Conn:    4 Mbps:     3257.876 Peak Mbps:     3341.420 Avg Mbps:      814.469
       5006      114901384      919.211   28.44%
       5006      112567984      900.544   27.86%
       5006       61887976      495.104   15.32%
       5006      114646592      917.173   28.38%
Conn:    4 Mbps:     3232.031 Peak Mbps:    3341.420 Avg Mbps:       808.008
```

Results from OpenBSD after intel card tuning

Increased hardware queues
TxDescriptors/RxDescriptors from 256 to 2048 in `src/sys/dev/pci/if_ix.h`

```
root@xpc02:root# tcpbench -n 4 10.0.10.30
 elapsed_ms           bytes           mbps   bwidth
...
Conn:    4 Mbps:     3292.587 Peak Mbps:     3345.610 Avg Mbps:      823.147
        4004        99490792      795.131   24.45%
        4004        98965000      790.929   24.32%
        4003       101455168      810.831   24.93%
        4003       107086560      855.837   26.31%
Conn:    4 Mbps:     3252.727 Peak Mbps:     3345.610 Avg Mbps:      813.182
        5006        99454488      794.048   24.63%
        5006        99545880      794.777   24.65%
        5006       102509992      818.443   25.38%
        5005       102365192      817.287   25.35%
Conn:    4 Mbps:     3224.555 Peak Mbps:    3345.610 Avg Mbps:       806.139
```

Ongoing project - not good enough yet

Tuning process performed on OpenBSD

Baseline testing - no optimizations

Update to latest kernel/driver, as recommended by misc@openbsd mailinglist

Update /etc/sysctl.conf with some tweaks

Update TxDescriptors/RxDescriptors

Monitor and watch using systat, vmstat etc. for sign of bottle-necks

Write misc@openbsd mailinglist - cry for help

Current OpenBSD configuration


```
/etc/sysctl.conf
ddb.panic=0                     # do not enter ddb console on kernel panic, reboot if possible
kern.bufcachepercent=90         # Allow the kernel to use up to 90% of the RAM for cache (default 10%)
machdep.allowaperture=2         # Access the X Window System (if you use X on the system)
net.inet.ip.forwarding=1        # Permit forwarding (routing) of packets through the firewall
net.inet.ip.ifq.maxlen=512      # Maximum allowed input queue length (256*number of physical interfaces)
net.inet.ip.mtudisc=0           # TCP MTU (Maximum Transmission Unit) discovery off since our mss is small enough
net.inet.tcp.rfc3390=2          # RFC3390 increasing TCP's Initial Congestion Window to 14600 for SPDY
net.inet.tcp.mssdflt=1440       # maximum segment size (1440 from scrub pf.conf match statement)
net.inet.udp.recvspace=231072 # Increase UDP "receive" buffer size. Good for 200Mbit without packet drop.
net.inet.udp.sendspace=231072 # Increase UDP "send" buffer size. Good for 200Mbit without packet drop.
```

and changes of TxDescriptors/RxDescriptors from 256 to 2048
in `src/sys/dev/pci/if_ix.h`

Troubleshooting and problems

Lost hardware - suddenly one XPC had 12Gb memory?

Leaking IP packets out through default gateway

Reverse lookups for non-existing addresses

Especially when working with hacker tools - take care!

Changes

To improve this setup I would:

- Create isolated network, firewall traffic from inside this network
- Add DNS server to isolated network
- Add network management server, which can measure on the network devices
- Add syslog server for catching kernel warnings etc.
- Configure the usual suspects Observium, RANCID etc.
  See http://files.kramse.org/tmp/osd-2011-Network-Management-print.pdf for inspiration
- Version control for configuration files, puppet or similar

Treat the setup as production, which can be tuned

Smokeping

When doing transfers across the internet, make sure to have Smokeping running

When you hear reports of slow transfers,
check your latency - increased latency $\rightarrow$ lower transfer speed

When you hear reports of slow transfers,
check your packet loss - increased packet loss $\rightarrow$ lower transfer speed

Important to keep historic data

`https://ring.nlnog.net/smokeping/solido01`

Conclusion

Open Source and 10Gbit is workable today on commodity hardware, Yay!

It currently does require some tuning to achieve high speeds

What about routing performance? Tomorrow and rest of week :-)

You MUST perform testing when you think the environment is ready

do NOT assume it works out of the box