Building Cost Effective High Performance 100 Gbps Firewalls

Jordan Alexis Caraballo-Vega, John E. Jasen
NASA Center for Climate Simulation
Goddard Space Flight Center

Objectives

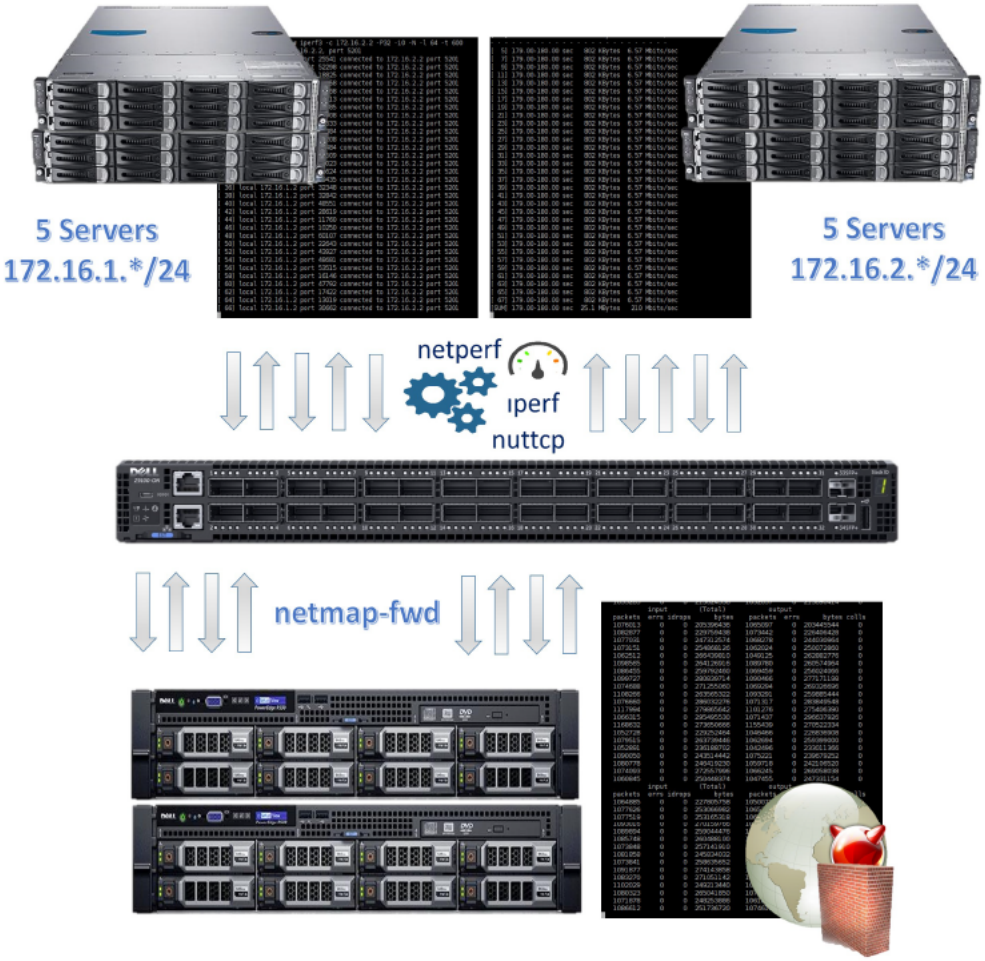
- Create a commodity based platform that can process enough packets per second (pps) to sustain a 100Gbps workload.
- Establish a test domain capable of sending massive amounts of pps in order to saturate the system.
- Determine operating system capable of processing more pps right out of the box.
- Define tuning variables needed to boost scores and decrease interrupts.

Background

- **FreeBSD Routing Capabilities**
  - FreeBSD systems are usually running less services, and packaged applications are configured by default with more performance tuning in mind.
- **Previous Work**
  - Previous work in the NCCS as a part of testing FreeBSD for high performance routing reached a maximum of 4 millions pps. Why Firewalls? More than 4000 nodes.
- **Netmap and Netmap-fwd**
  - A framework for high speed packet I/O implemented as a single, non intrusive kernel module. Together with netmap-fwd API can easily reach line rate on 10G NIC's.

Hardware - Systems

- **Clients**
  - 4 Dell C60100 Servers
  - 1 Dell 420 Server
- **Servers**
  - 4 Dell C6100 Servers
  - 1 Dell 420 Server
- **Router**
  - Dell 530 Server



Hardware – NIC's

- **Clients and Servers**
  - 10G Intel 8259x
  - 10G Mellanox ConnectX2
- **Router**
  - 40G Mellanox ConnectX4
  - 40G Chelsio T580

Network Configurations

- **Ip addresses range**

  # ifconfig_ix0=”inet 172.16.2.$IP netmask 255.255.255.0 mtu 2000 -lro -tso up”

  # static_routes=”test”

  # route_test=”-net 172.16.1.0/24 172.16.2.1”

- **Gateway Configuration**

```
hostname="test-router-1"
ifconfig_bge0="DHCP"
ifconfig_bge0_ipv6="inet6 accept_rtadv"

# Ethernet FortyG Interfaces
ifconfig_cxl0="inet 172.16.1.1 netmask 255.255.255.0 mtu 9000 -lro -tso up"
ifconfig_cxl1="inet 172.16.2.1 netmask 255.255.255.0 mtu 9000 -lro -tso up"

gateway_enable="YES"
sshd_enable="YES"
# Set dumpdev to "AUTO" to enable crash dumps, "NO" to disable
dumpdev="AUTO"
```

Router Configurations

- **Tuning Files**

  /etc/rc.conf and
  /boot/loader.conf

- **Tuning Variables**
  - Number of rx and tx ques
  - Rx and tx ques size
  - Disable pauses and flowcontrol
  - Buffers size

```
# maxthreads appears to not exceed CPU - applied by John Jasen
net.isr.maxthreads=18

# bindthreads may be indicated when using cpuset(1) on interrupts - applied by John Jasen
net.isr.bindthreads=1

# random guess based on googling - Applied by John Jasen
net.isr.maxqlimit=60480
#net.link.ifqmaxlen=90000
net.link.ifqmaxlen=16384

# hostcache cachelimit is the number of ip addresses in the hostcache list.
# Setting the value to zero(0) stops any ip address connection information from
# being cached and negates the need for "net.inet.tcp.hostcache.expire".
# from https://calomel.org/freebsd_network_tuning.html
net.inet.tcp.hostcache.cachelimit="0"

# Increasing number of ques
dev.cxl.0.nrxq=32
dev.cxl.1.nrxq=32
dev.cxl.0.ntxq=32
dev.cxl.1.ntxq=32

# rx and tx size
dev.cxl.0.qsize_txq=8192
dev.cxl.0.qsize_rxq=8192
dev.cxl.1.qsize_txq=8192
dev.cxl.1.qsize_rxq=8192

# The number of TOE tx queues to use for a 10Gb or 40Gb port.  The
# default is 8 or the number of CPU cores in the system, whichever
# is less. From cxgbe page
dev.cxl.1.nofldtxq=16
dev.cxl.1.nofldrxq=8
dev.cxl.0.nofldtxq=16
dev.cxl.0.nofldrxq=8

# drop toecaps to increase queues
dev.t5nex.0.toecaps=0
dev.t5nex.0.rdmacaps=0
dev.t5nex.0.iscsicaps=0
```

Router Configurations

- **Balancing load interrupts**

  Script to move interrupts to the CPU's closest to the PCI.

```
#!/usr/local/bin/bash
ncpu=36
#irqlist=$(vmstat -ia | egrep 'mlx' | cut -f1 -d: | cut -c4-)
irqlist=$(vmstat -ia | egrep 't5nex0' | cut -f1 -d: | cut -c4-)
i=18
for irq in $irqlist; do
        echo cpuset -l $i -x $irq
        cpuset -l $i -x $irq
        i=$((i+1))
        [ $i -ge $ncpu ] && i=18
done
```

```
last pid: 17951;  load averages:  9.24,  7.26,  5.33
248 processes: 44 running, 126 sleeping, 78 waiting
CPU 0:   0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100% idle
CPU 1:   0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100% idle
CPU 2:   0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100% idle
CPU 3:   0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100% idle
CPU 4:   0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100% idle
CPU 5:   0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100% idle
CPU 6:   0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100% idle
CPU 7:   0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100% idle
CPU 8:   0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100% idle
CPU 9:   0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100% idle
CPU 10:  0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100% idle
CPU 11:  0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100% idle
CPU 12:  0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100% idle
CPU 13:  0.0% user,  0.0% nice,  0.4% system,  0.0% interrupt, 99.6% idle
CPU 14:  0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100% idle
CPU 15:  0.0% user,  0.0% nice,  0.4% system,  0.0% interrupt, 99.6% idle
CPU 16:  0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100% idle
CPU 17:  0.0% user,  0.0% nice,  0.0% system,  0.0% interrupt,  100% idle
CPU 18:  0.0% user,  0.0% nice,  0.0% system, 29.3% interrupt, 70.7% idle
CPU 19:  0.0% user,  0.0% nice,  0.0% system, 27.4% interrupt, 72.6% idle
CPU 20:  0.0% user,  0.0% nice,  0.0% system, 74.8% interrupt, 25.2% idle
CPU 21:  0.0% user,  0.0% nice,  0.0% system, 78.9% interrupt, 21.1% idle
CPU 22:  0.0% user,  0.0% nice,  0.0% system, 81.9% interrupt, 18.1% idle
CPU 23:  0.0% user,  0.0% nice,  0.0% system, 81.9% interrupt, 18.1% idle
CPU 24:  0.0% user,  0.0% nice,  0.0% system, 73.7% interrupt, 26.3% idle
CPU 25:  0.0% user,  0.0% nice,  0.0% system, 81.1% interrupt, 18.9% idle
CPU 26:  0.0% user,  0.0% nice,  0.0% system, 80.0% interrupt, 20.0% idle
CPU 27:  0.0% user,  0.0% nice,  0.0% system,  100% interrupt,  0.0% idle
CPU 28:  0.0% user,  0.0% nice,  0.0% system,  0.4% interrupt, 99.6% idle
CPU 29:  0.0% user,  0.0% nice,  0.0% system,  0.4% interrupt, 99.6% idle
CPU 30:  0.0% user,  0.0% nice,  0.0% system, 33.0% interrupt, 67.0% idle
CPU 31:  0.0% user,  0.0% nice,  0.0% system, 25.9% interrupt, 74.1% idle
CPU 32:  0.0% user,  0.0% nice,  0.0% system, 34.4% interrupt, 65.6% idle
CPU 33:  0.0% user,  0.0% nice,  0.0% system, 33.0% interrupt, 67.0% idle
CPU 34:  0.0% user,  0.0% nice,  0.0% system, 33.0% interrupt, 67.0% idle
CPU 35:  0.0% user,  0.0% nice,  0.0% system, 33.0% interrupt, 67.0% idle
Mem: 8300K Active, 197M Inact, 1662M Wired, 295M Buf, 29G Free
Swap: 3881M Total, 3881M Free
```

Tools and Test Configuration

- **Active Measurements**

  Iperf3, Nuttcp, Pkt-gen

- **Monitoring Tools**

  Netstat, Top, PMC

- Type

  TCP, UDP

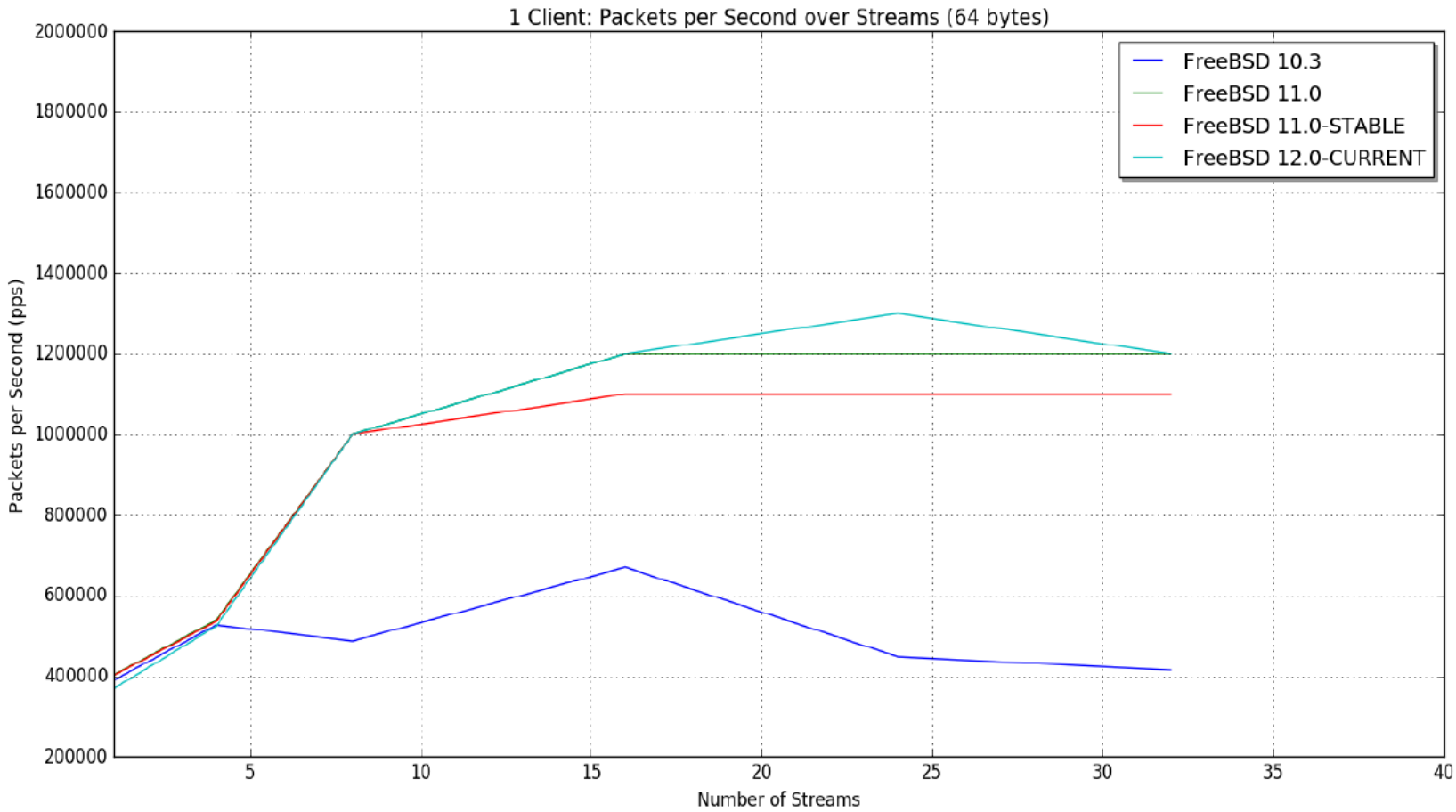
- Packet size

  64, 1500, 9000 mtu

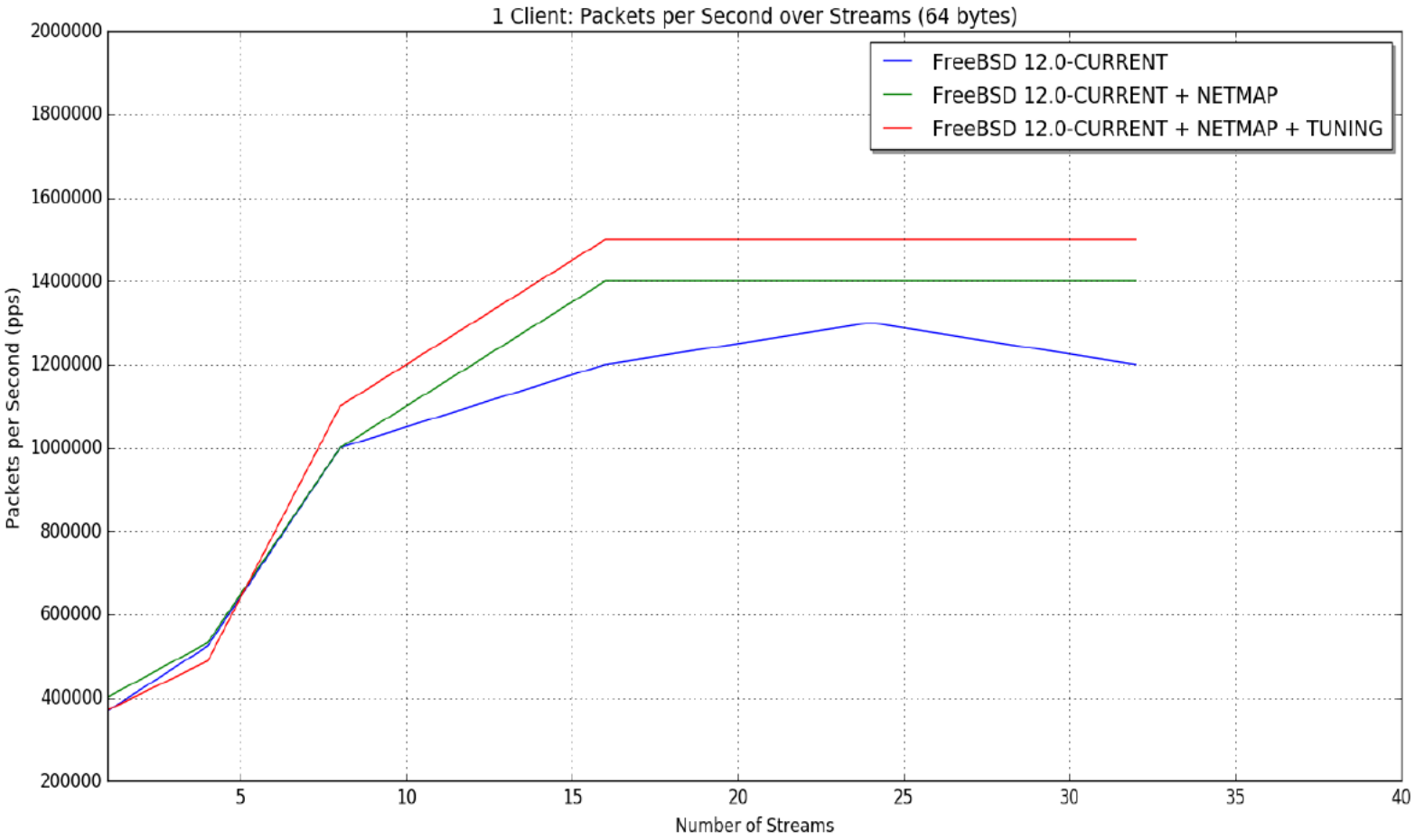
```
# Global Variables
TEST=tcp
IP=6
PKT=64
TIME=6000
STREAMS=10
DEFPORT=5600
PORTS=80

# Iterate through ports and start tests
if [ $TEST == "tcp" ]; then
    for ((i=1;i<=PORTS;i++)); do
        PORT=$(($DEFPORT+$i))
        iperf3 -c 172.16.2.$IP -i0 -N -l$PKT -t$TIME -P$STREAMS -p$PORT &
    done
else
    for ((i=1;i<=PORTS;i++)); do
        PORT=$(($DEFPORT+$i))
        iperf3 -c 172.16.2.$IP -u -b10000m -i 0 -N -l$PKT -t$TIME -P$STREAMS -p$PORT &
    done
fi
```

```
          input        (Total)           output
packets  errs idrops      bytes    packets  errs      bytes colls drops
   2.7M     0  191k       2.2G       2.5M     0       2.0G     0     0
   2.7M     0  193k       2.2G       2.5M     0       2.0G     0     0
   2.7M     0  190k       2.2G       2.5M     0       2.0G     0     0
   2.7M     0  190k       2.2G       2.5M     0       2.0G     0     0
   2.7M     0  189k       2.2G       2.5M     0       2.0G     0     0
   2.7M     0  192k       2.2G       2.5M     0       1.9G     0     0
   2.7M     0  196k       2.2G       2.6M     0       2.0G     0     0
   2.7M     0  195k       2.2G       2.5M     0       2.0G     0     0
   2.7M     0  195k       2.2G       2.5M     0       2.0G     0     0
   2.8M     0  197k       2.3G       2.6M     0       2.0G     0     0
   2.7M     0  190k       2.2G       2.5M     0       2.0G     0     0
   2.7M     0  197k       2.3G       2.5M     0       2.0G     0     0
   2.7M     0  195k       2.2G       2.5M     0       2.0G     0     0
   2.7M     0  198k       2.2G       2.5M     0       2.0G     0     0
   2.7M     0  194k       2.3G       2.5M     0       2.0G     0     0
   2.7M     0  199k       2.3G       2.5M     0       2.0G     0     0
   2.7M     0  197k       2.2G       2.5M     0       2.0G     0     0
   2.7M     0  194k       2.2G       2.5M     0       2.0G     0     0
   2.7M     0  193k       2.2G       2.5M     0       2.0G     0     0
   2.7M     0  194k       2.3G       2.5M     0       2.0G     0     0
   2.7M     0  197k       2.3G       2.5M     0       2.0G     0     0
```

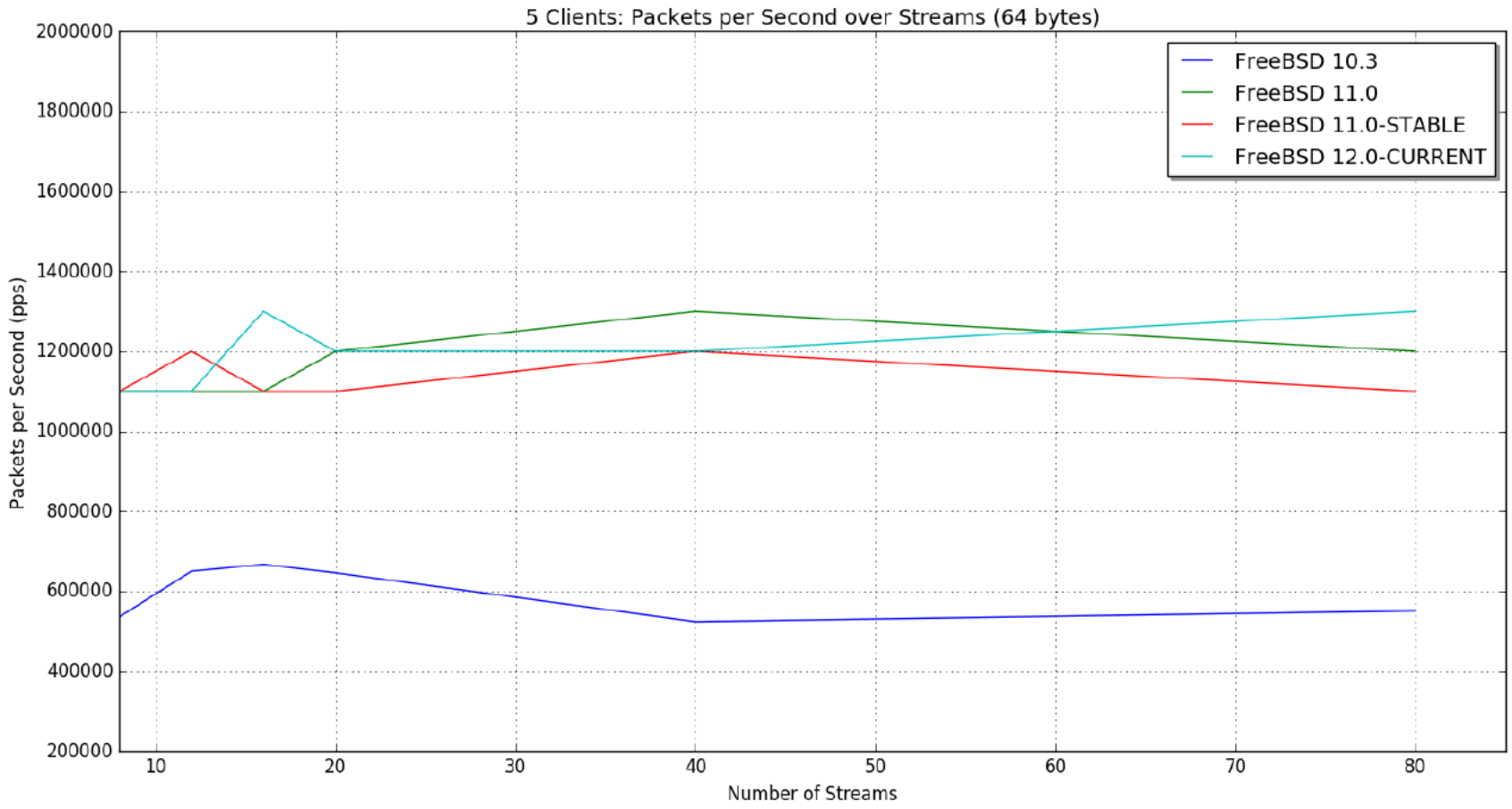
Results


1 Client: Packets per Second over Streams (64 bytes)
Packets per Second (pps)
Number of Streams
2000000
1800000
1600000
1400000
1200000
1000000
800000
600000
400000
200000
5
10
15
20
25
30
35
40
FreeBSD 10.3
FreeBSD 11.0
FreeBSD 11.0-STABLE
FreeBSD 12.0-CURRENT

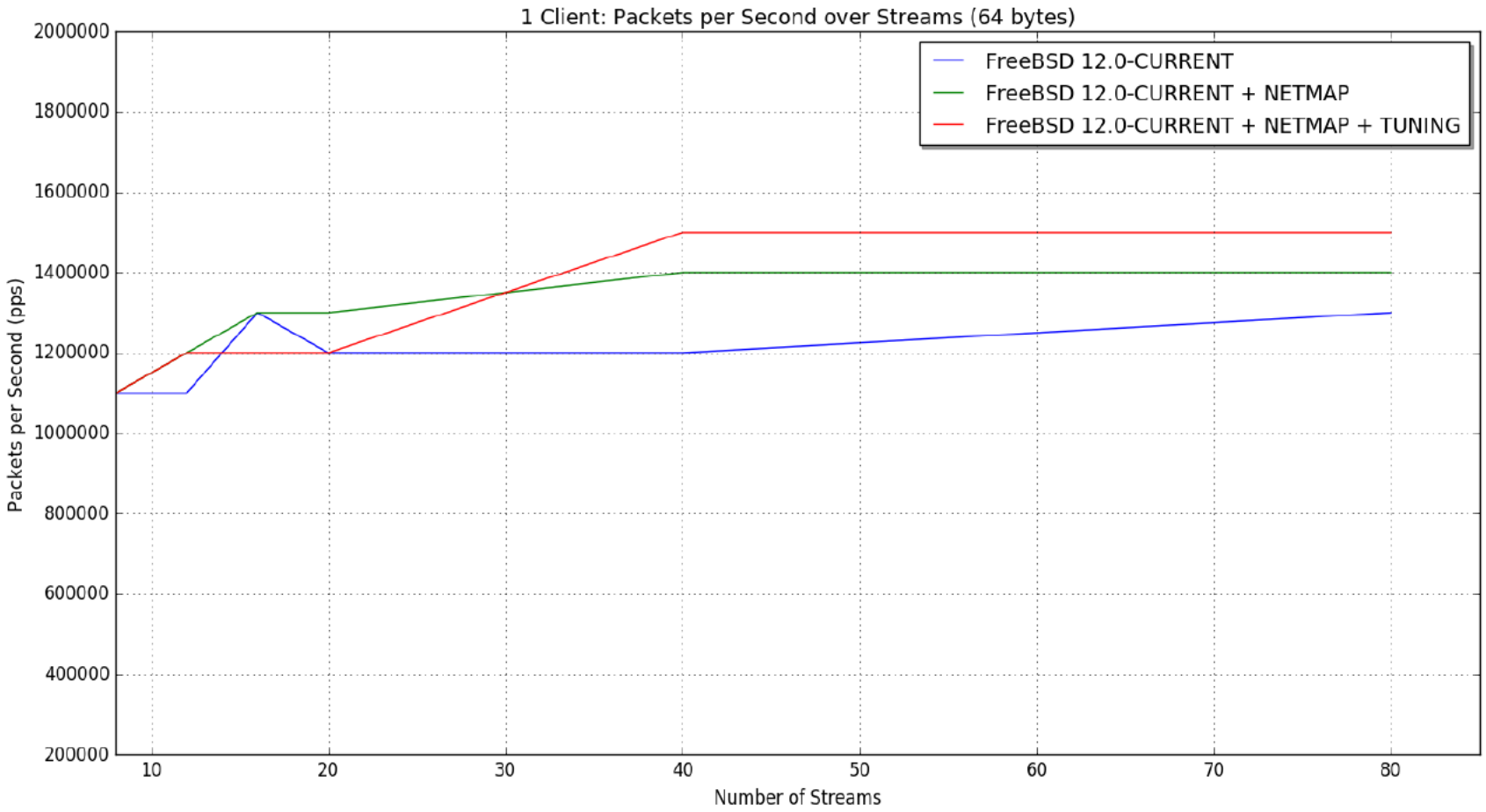
Results


1 Client: Packets per Second over Streams (64 bytes)
FreeBSD 12.0-CURRENT
FreeBSD 12.0-CURRENT + NETMAP
FreeBSD 12.0-CURRENT + NETMAP + TUNING
Packets per Second (pps)
2000000
1800000
1600000
1400000
1200000
1000000
800000
600000
400000
200000
5
10
15
20
25
30
35
40
Number of Streams

Results


5 Clients: Packets per Second over Streams (64 bytes)
FreeBSD 10.3
FreeBSD 11.0
FreeBSD 11.0-STABLE
FreeBSD 12.0-CURRENT
Packets per Second (pps)
2000000
1800000
1600000
1400000
1200000
1000000
800000
600000
400000
200000
10
20
30
40
50
60
70
80
Number of Streams

Results


1 Client: Packets per Second over Streams (64 bytes)
FreeBSD 12.0-CURRENT
FreeBSD 12.0-CURRENT + NETMAP
FreeBSD 12.0-CURRENT + NETMAP + TUNING
Packets per Second (pps)
2000000
1800000
1600000
1400000
1200000
1000000
800000
600000
400000
200000
10
20
30
40
50
60
70
80
Number of Streams

Interesting Features

- Testing has shown that the amount of pps will rise as the newer versions of the operating system have been deployed.
- FreeBSD was able to send more pps as client than Centos 6.
- The choice of network card can have a significant impact on pps, tuning, and netmap support.
- Increasing the amount of streams and ports through time drops the packet amount. About 10 streams and 80 ports per client achieved the best scores.
- Netmap-fwd increases the amount of packets significantly.

What is next?

- Further test is needed.
- Enable the system with two 40G cards.
- Configure Lagg in the router switch ports.
- Updates to the netowrk capabilities in the FreeBSD-Current version will be closely monitored and applied as appropiate.
- The final result will be a reference architecture with representative hardware and software that will enable the NCCS build, deploy, and efficiently maintain extremely cost-effective 100Gbps firewalls.

References

- FreeBSDforwardingPerformance. RetrievedonAugust22, 2016 fromhttps://bsdrp.net/documentation/technical_docs/performance
- ProjectRoutingProposal. RetrievedonSeptember18, 2016 fromhttps://wiki.freebsd.org/ProjectsRoutingProposal/ConversionStatus#New_routing_KPI_conversion_status
- Netmap. RetrievedonAugust22,2016 from https://github.com/luigirizzo/netmap
- Scaptest. RetrievedonAugust22,2016 from https://github.com/jordancaraballo/scaptest