

Welcome to

## A Smart OpenBSD CPE

Henrik Lund Kramshøj  
hlk@pasientsky.no  
hlk@patientsky.com

Slides are available as PDF [kramshoej@Github](mailto:kramshoej@Github)



Not affiliated with the OpenBSD project, but a long time and very happy user

# Goal

PatientSky is rolling out a new health infrastructure for clinics in Norway.

- OpenBSD as a CPE in a network with ordinary internet traffic and VoIP
- Juniper MX and OpenBSD in the datacenter, all firewalls are OpenBSD
- BGP, PF and service daemons
- Juniper configuration and OpenBSD configs
- BGP to PF Tables, firewalling/NAT based on BGP updates
- OpenBSD niceness, why choosing OpenBSD made a lot of things easier
- Keywords: OpenBSD, BGP, routing, IEEE 802.1q, VLAN, IEEE802.1p, CoS/QoS, VoIP, firewalling, JSON config

I really can't explain everything in 40min, so ask us questions after

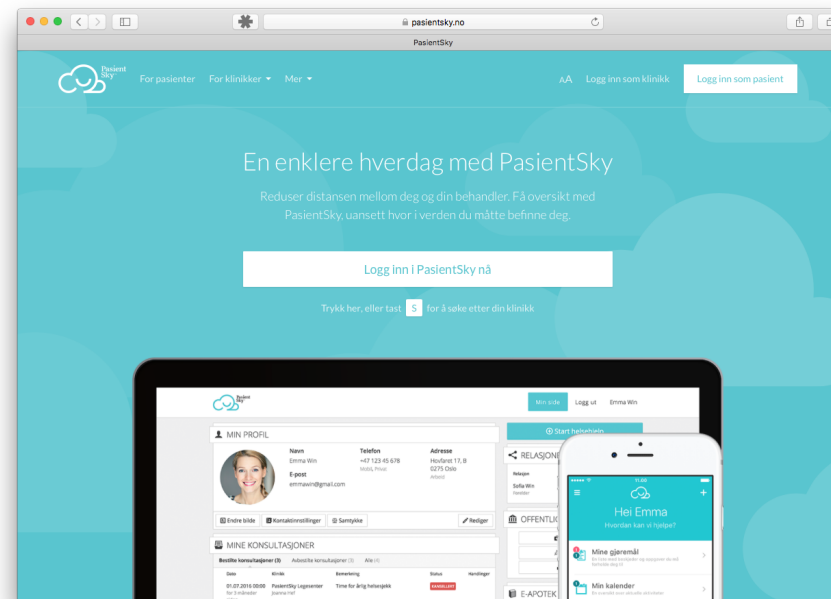
# Thank you OpenBSD

Before I get started I need to say hi, and a big thank you to Peter Hessler I have had the pleasure of help from him getting all this working!

I am also sure he can interrupt me if I say something wrong or stupid about OpenBSD, OpenBGP and the rest. 😊Peter, if you need a drink today, just say so to me or my boss Andreas!

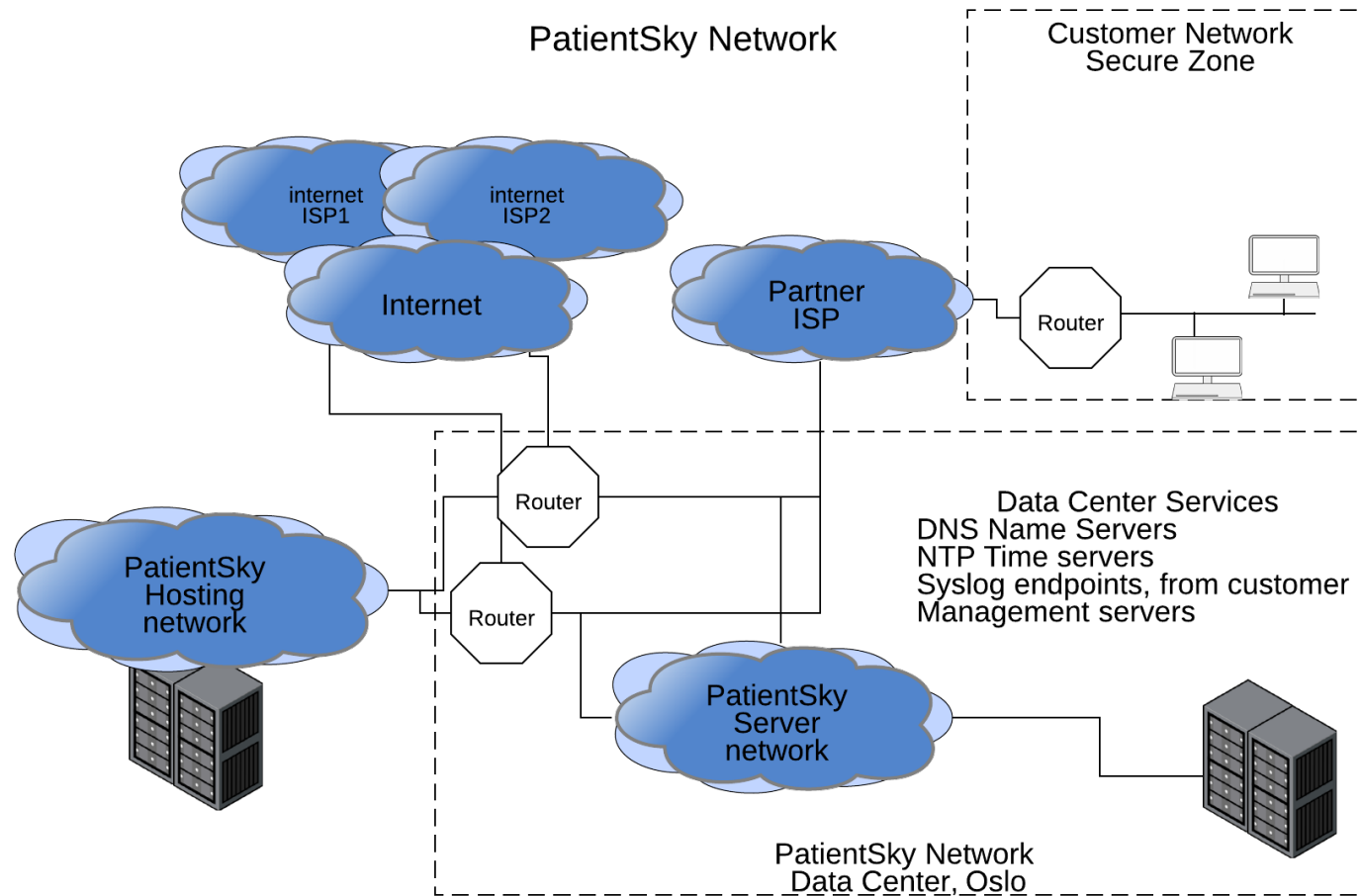


# Pasientsky.no - the environment and services

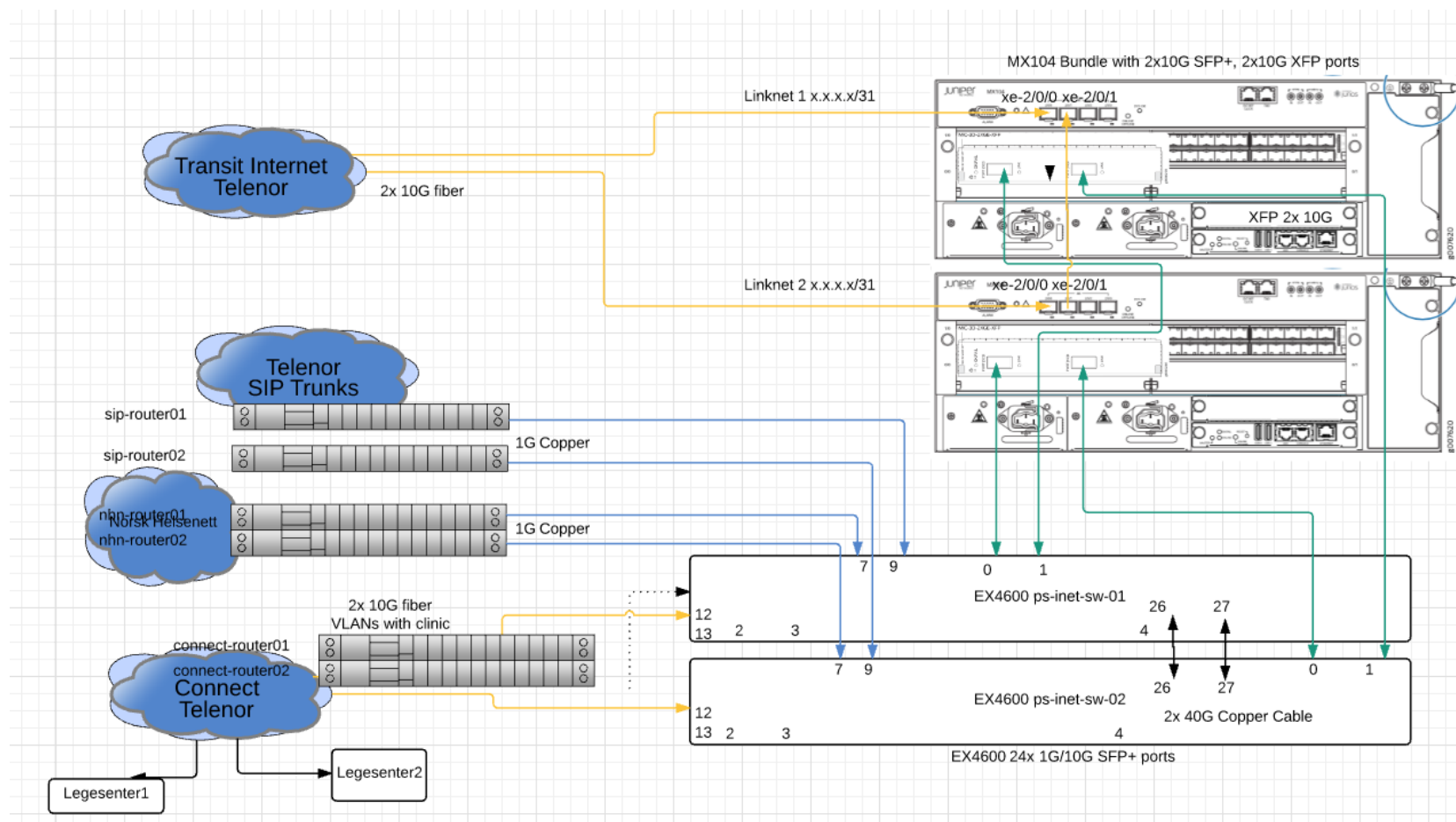


Connected Clinic from PasientSky provides modern and revolutionary solutions meeting the special communication needs in health care. A small and smart box provides quick and stable fiber internet connection with integrated telephony in browser.

# Overview

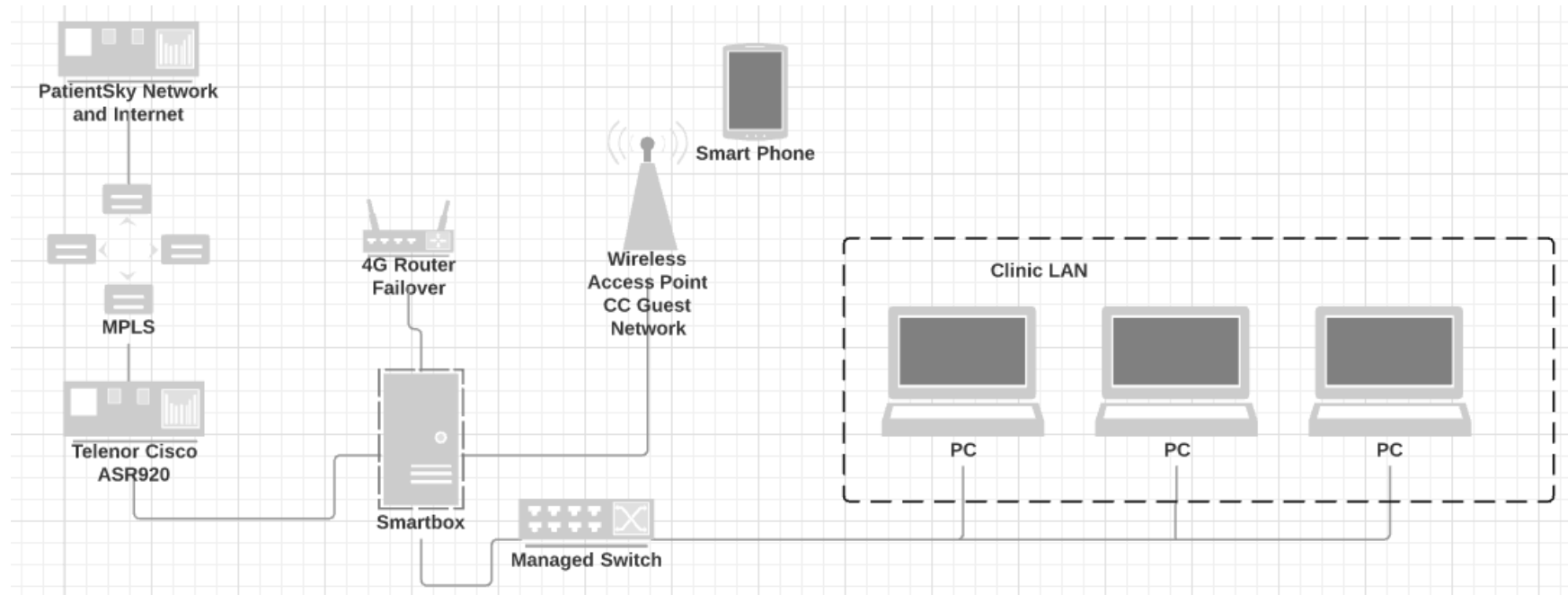


# Core Network - Juniper



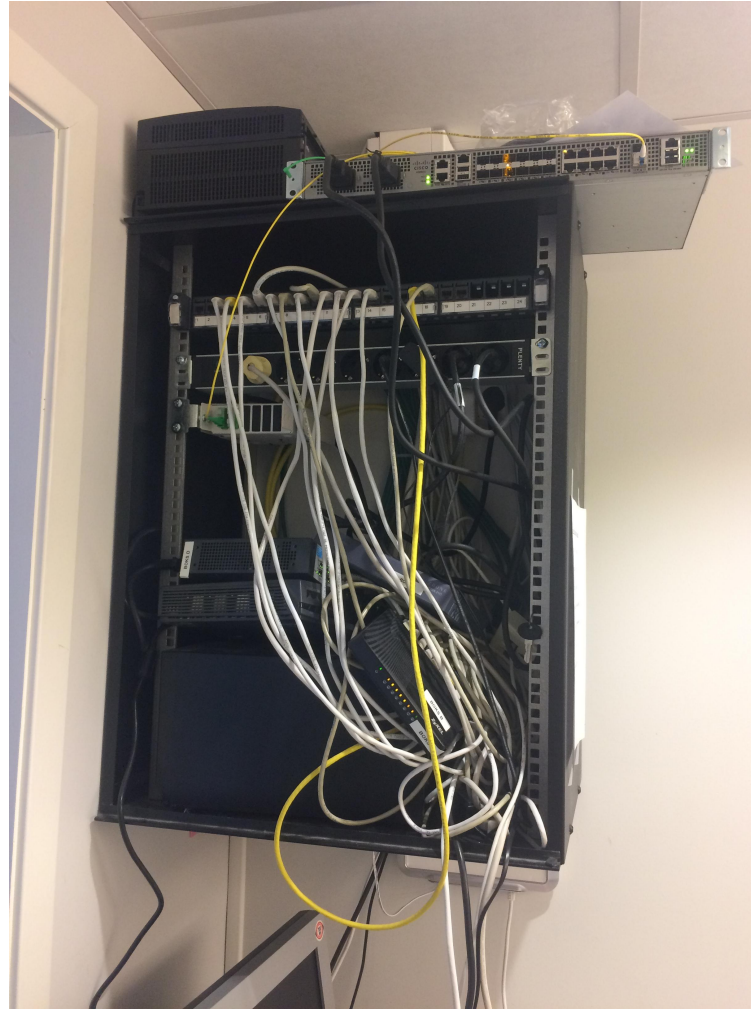
Mostly 10G links backhauling Ethernet Connect Layer 2 into our network, provides Transit and Norsk Helsenett (National Health care network)

# OpenBSD CPE: BGP, PF and service daemons



- Soekris Net6501-50 1 Ghz CPU, 1024 Mbyte DDR2-SDRAM, 4 x 1Gbit Ethernet
- OpenBSD operating system
- Solid hardware + free operating system = reliable service
- Yes, Telenor uses ASR920 as CPE for 8Mbit SHDSL too☺

## Existing infrastructure, in some places



Pretty nice heh, not always this bad, but usually not good



## Current usage stats from a CPE:

top output:

```
load averages:  0.14,  0.09,  0.08                smartbox-xxx-01 11:19:15
37 processes: 36 idle, 1 on processor                up 106 days,  3:49
CPU0 states:  0.0% user,  0.0% nice,  0.2% system,  0.0% interrupt, 99.8% idle
CPU1 states:  1.0% user,  0.0% nice,  0.4% system,  0.4% interrupt, 98.2% idle
Memory: Real: 34M/225M act/tot Free: 757M Cache: 110M Swap: 0K/2048M
```

```
# bgpctl show
```

| Neighbor       | AS    | MsgRcvd | MsgSent | OutQ | Up/Down  | State/PrfRcvd |
|----------------|-------|---------|---------|------|----------|---------------|
| 185.161.12xx.x | 50033 | 105419  | 67948   | 0    | 01w2d08h | 9754          |

```
# wc -l /etc/pf.conf
      86 /etc/pf.conf
```

```
root@smartbox-xxx-01:root# grep -v "^#" /etc/sysctl.conf
net.inet.ip.forwarding=1
ddb.panic=0
```

Sorry, no IPv6 yet, my fault, already configured in data center interfaces

# Important processes and components

- OpenBSD kernel - does routing, thank you
- OpenBSD kernel - multiple routing tables, allow drop-in replacement in networks
- OpenNTP - time keeping
- OpenBGP - BGP to get NHN prefixes
- relayd - provides failover, change default route
- OpenSSH - secure remote access
- DHCPD - dhcp service to LAN
- OpenBSD PF - awesome firewall connecting it all nicely
- OpenBSD PF queue - allow detailed control of bandwidth
- OpenBSD PF prio into VLAN header - QoS/CoS of VoIP traffic
- Python - we have a few scripts to configure the above with templates from single JSON config

# Relayd failover to 4G router

```
# cat /etc/relayd.conf
primary = "185.161.xxx.x"
secondary = "192.168.8.1"
interval 10
table <gateways> { $primary ip ttl 1 priority 10, $secondary ip ttl 1 priority 50 }
router "uplinks" {
    route 0.0.0.0/0
    forward to <gateways> check icmp
}
```

Easy to understand, easy to implement

# OpenBSD queue pf.conf - from 20/20Mbit customer

```
# Queue to fix TCP originating from our smartbox, if we send more than
# bandwidth the shaping done by Telenor cause huge backoffs
queue root on em3 bandwidth 20M max 20M

# Currently used for VoIP and PatientSky Hosting
# Note: VoIP Max 25% of bandwidth, excess dropped by provider!
queue high parent root bandwidth 5M max 5M
queue normal parent root bandwidth 20M max 20M default
queue low parent root bandwidth 15M max 15M

# Download queues on inside interface to LAN
# by limiting this, we end up receiving less than max from outside
queue dn_parent on em2 bandwidth 20M max 20M
queue dn_high parent dn_parent bandwidth 20M max 20M
queue dn_default parent dn_parent bandwidth 15M max 15M default

# Wifi
queue guest_parent on em0 bandwidth 15M max 15M
queue guest_default parent guest_parent bandwidth 15M max 15M default
```

You can only limit what you send, download queues remove need for specific queue in data center for EACH customer!

# OpenBSD use queueing in rules

```
table <HOSTED_NETWORKS> const { 185.60.160.0/22 }

# Rules start here
block all

# Normal would be 3 and patientsky higher priority
pass out set queue normal set prio 3
pass out to <HOSTED_NETWORKS> set queue high set prio 5

# High prio on all traffic originating from us and our <HOSTED_NETWORKS> address space
pass in on egress from <HOSTED_NETWORKS> to (egress:0) set queue dn_high set prio 5
```

- When you limit outgoing - to the LAN, results is because of TCP it limits what you receive :-)
- Not really doing queuing inside LAN, some switches not controlled, customer responsibility
- If internal LAN with gigabit switches cannot handle VoIP, expect other problems

# OpenBGP download prefixes to PF Tables

```
...  
# neighbors and peers  
psnet="50033"  
group "nhn" {  
# peering to get NHN network  
    remote-as $psnet  
    neighbor 185.161.xxx.xx  
}  
  
# Our local network  
network 172.22.xxx.0/27  
  
# Filtering  
allow from any  
allow from AS 50033  
match from group "nhn" community $psnet:56828 set pftable "NHN"
```

Two functions, announce our local NHN prefix, internal LAN IP  
and getting a table of almost 10.000 prefixes

# OpenBSD multiple routing domains are cool

with BGP running we can use the prefixes in rules, here no-NAT rule:

```
# towards end of pf.conf
# Routing Domain 1 used for LAN
anchor "inside" on rdomain 1 {
    # Allow administrative access when on-site
    pass in quick on em2 inet proto tcp from any to em2 port 34
    # Internal LAN must be allowed out
    pass in on em2
    # Guest network, no access to internal LAN or NHN
    # Prio 0 in Telenor is Best Effort
    pass in on em0 to { !(em2:network) !<NHN> } set queue low set prio 0
    # Make sure our Hosted networks have priority and NHN traffic is sent through unharmed
    pass out quick to <HOSTED_NETWORKS> nat-to (egress:0) rtable 0 set queue high set prio 5
    pass out quick to <NHN> rtable 0 set queue normal set prio 3
    pass out to !<INSIDE_NETWORKS> nat-to (egress:0) rtable 0 set queue normal set prio 3
}
```

and check using:

```
echo "Checking NHN DNS from routing table"
NHNIP=`ifconfig em2 | grep inet | cut -f 2 -d ' '`
route -T 1 exec dig @172.21.1.2 +short -b $NHNIP smtp.nhn.no || exit 1
```

# OpenBSD priority

```
pass out quick to <HOSTED_NETWORKS> nat-to (egress:0) rtable 0 set queue high set prio 5
```

```
pass in proto tcp to port 25 set prio 2
```

```
pass in proto tcp to port 22 set prio (2, 5)
```

Prio is copied directly into IEEE 802.1q header, making it easy to use IEEE 802.1p

If the packet is transmitted on a vlan(4) interface, the queueing priority will also be written as the priority code point in the 802.1Q VLAN header. If two priorities are given, packets which have a TOS of lowdelay and TCP ACKs with no data payload will be assigned to the second one.

Hint: OpenSSH `sshd_config` using IPQoS can achieve the same



# Junos MX config, show configuration class-of-service

```
rewrite-rules {
  ieee-802.1 telenor {
    forwarding-class assured-forwarding {
      loss-priority low code-point 101;
      loss-priority high code-point 101;
      loss-priority medium-high code-point 101;
      loss-priority medium-low code-point 101;
    }
    forwarding-class expedited-forwarding {
      loss-priority low code-point 011;
      loss-priority high code-point 011;
      loss-priority medium-high code-point 011;
      loss-priority medium-low code-point 011;
    }
  }
}
```

Pro tip: this requires traffic to already be classified into these classes.  
We solved it by sending VLAN traffic with prio from OpenBSD in data center to MX

## Junos outgoing, show configuration class-of-service

```
interfaces {
  ae0 {
    ...
    unit 1008 {
      classifiers {
        ieee-802.1 default;
      }
      rewrite-rules {
        ieee-802.1 telenor vlan-tag outer-and-inner;
      }
    }
  }
}
```

Note: We use double vlan-tags outer 1008 inner 100 in data center.  
This ends up with VLAN 100 on ALL smartboxes/sites

Result: We have the same simple config on all smartboxes

# OpenBSD niceness

Why choosing OpenBSD made a lot of things easier

- Free to install routers, firewalls, where we need them, no license
- Secure and stable, less worries, stable network yay!
- Nifty tricks with OpenBGP makes for a very elegant PF config
- PF integrated with IEEE 802.1p - on VLAN interfaces
- PF has a very readable format with syntactic sugar and dynamic constructs like `(em2:network)` the network on interface em2
- OpenBSD has stable release schedule, every 6 months

TL;DR Full control with easy transparent configs

# OpenBSD CPE JSON config

```
{
  "system": {
    "config-version": "1.0",
    "nameserver1": "185.161.125.241",
    "nameserver2": "185.161.127.241",
    "ntpserver1": "185.161.125.241",
    "ntpserver2": "185.161.127.241",
    "firmware-version": "590",
    "hostname": "smartbox-xxx-01",
    "package-repository": "http://...",
    "update-server": "http://...",
    "guest-network": "false"
  },
  "network-primary": {
    "gateway": "185.161.12x.xxx",
    "ipaddress": "185.161.12x.xxy",
    "subnet-mask": "255.255.255.248",
    "vlan": "100",
    "bandwidth": "20M"
  },
  "network-secondary": {
    "enabled": "false",
    "gateway": "192.168.8.1",
    "ipaddress": "192.168.8.10",
    "subnet-mask": "255.255.255.0"
  },
  "lan": {
    "ipaddress": "172.22.xxx.1",
    "subnet-mask": "255.255.255.224",
    "local-nhn": "172.22.xxx.0/27"
  },
  "dhcp": {
    "enabled": "true",
    "domain-name": "patientsky.com",
    "network": "172.22.xxx.0",
    "range": "172.22.xxx.3 172.22.xxx.30",
    "subnet-mask": "255.255.255.224"
  },
  "bgp": {
    "enabled": "true",
    "neighbor": "185.161.1xx.xxx"
  }
}
```

Python tool: pxeboot, ./sbimport conf/smartbox.conf && reboot

Custom config of PF on some sites, currently using Ansible template push

## Conclusion

No errors or obstacles - no road blocks

Working as intended, great!

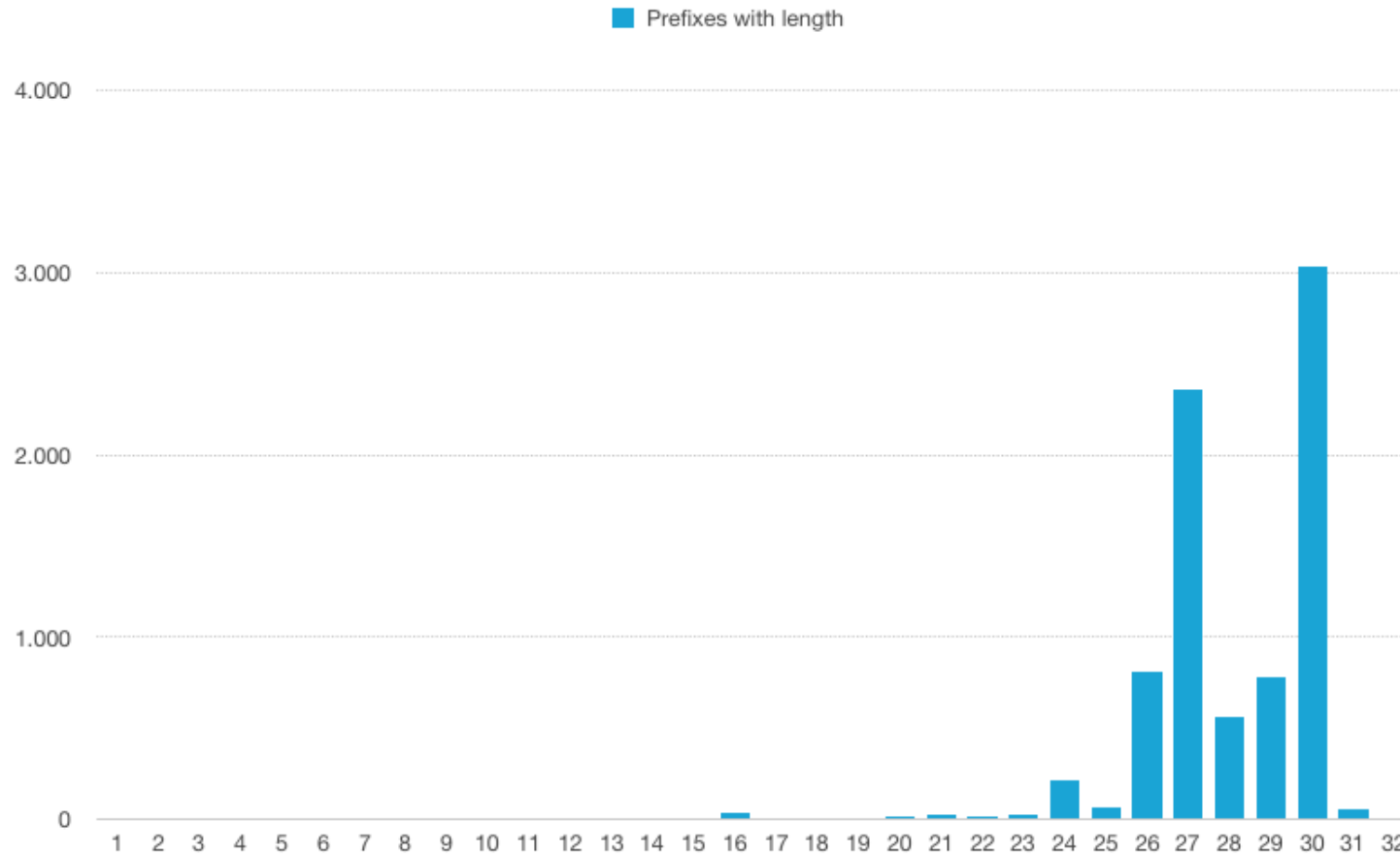
Almost all parts are in OpenBSD base!

## OpenBSD is here already - use it



Logo from <http://www.openbgpd.org/>

# NHN BGP routing table - 10K entries



Sure, put 3.000 prefixes with length /30 into the table, linknets?