The journey to Opensource networking with OpenBSD @AS15693

Introduction

Wouter Prins

Network Designer, AS15693, BusinessConnect BV

Network Designer at KPN, AS1136 via Routz

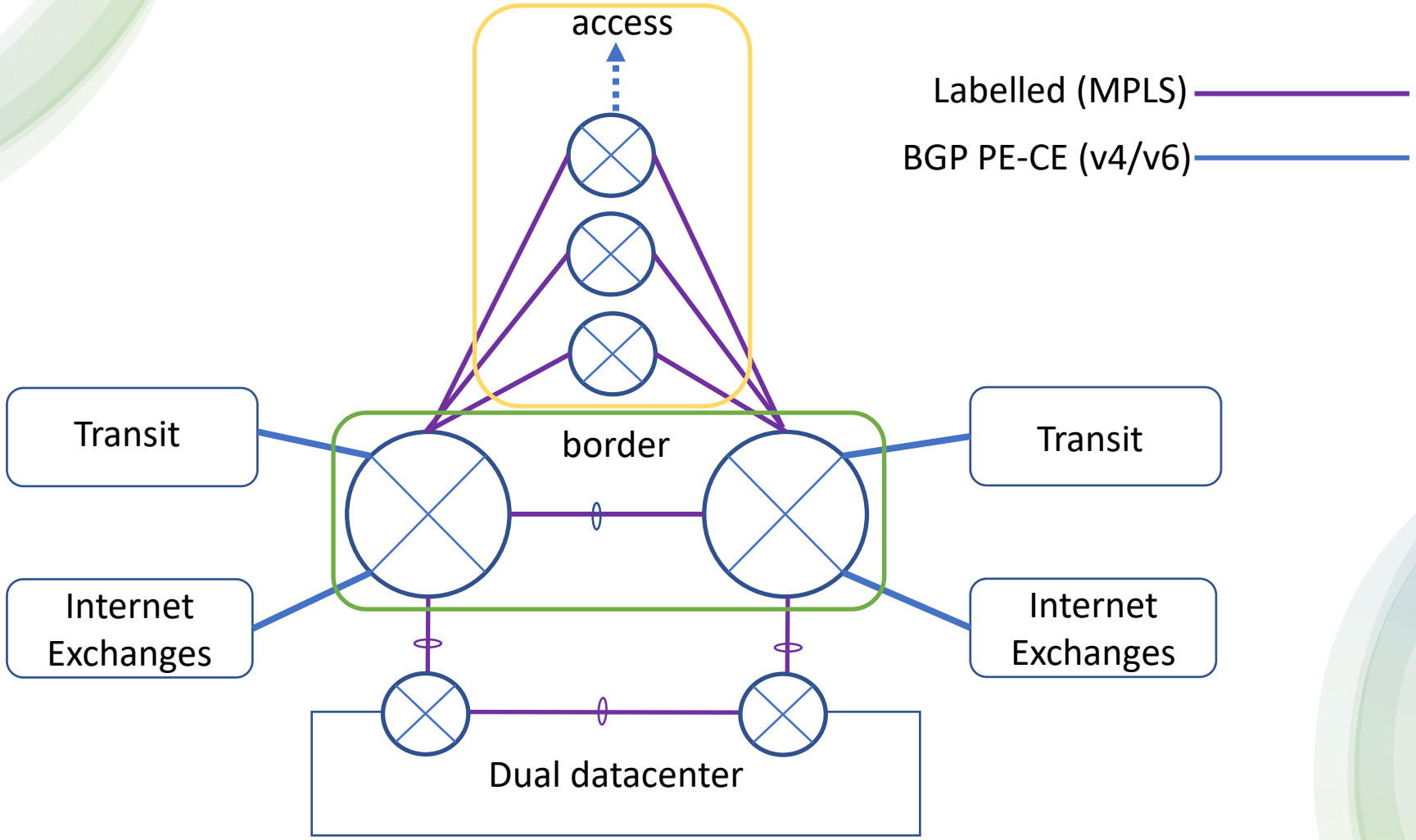
Experience:

~10 years of experience with FreeBSD

>20 years of experience with Linux

>20 years of experience in networking with multiple vendors

Network topology and layers



Multivendor network: Cisco (multiple series), Brocade VDX, Juniper MX series

The challenge in 2021

- Juniper MX104 border routers using multiple full-tables (IPv4 and IPv6)
- Slow CPU (PowerPC), 32 bits OS and too many tasks to fulfill
- Memory leak issues with 4G RAM
- Lack of 10G interfaces and workarounds
- New hardware based router platforms are too expensive

Research on (cheaper) alternatives

- Using x86 servers, either bare-metal or virtualization or a combination of it
- Network Interface Cards and *wdm/grey optics compatibility
- Opensource OS, either Linux or *BSD based
  - Important topics
- Security and hardening
- Network requirements

Network Requirements

- OSPF as an IGP for IPv4 and IPv6
- BGP: IPv4- and IPv6 unicast, VPNv4 and VPNv6 support
- BGP: Inband route-reflector
- BGP: Support for multiple full-tables (IPv4/IPv6)
- BGP: RPKI Route Origin Validation support, RTR support
- MPLS applications: L3VPN and L2VPN, using LDP
- VRF-Lite for out-of-band management
- PPPoA/PPPoE termination
- 1G or 10G ports and optics (SX/LX, SR/LR), LACP
- 15-20 Gb/s (internal- and external traffic)
- Netflow/IPFix support

Results of the research and choices

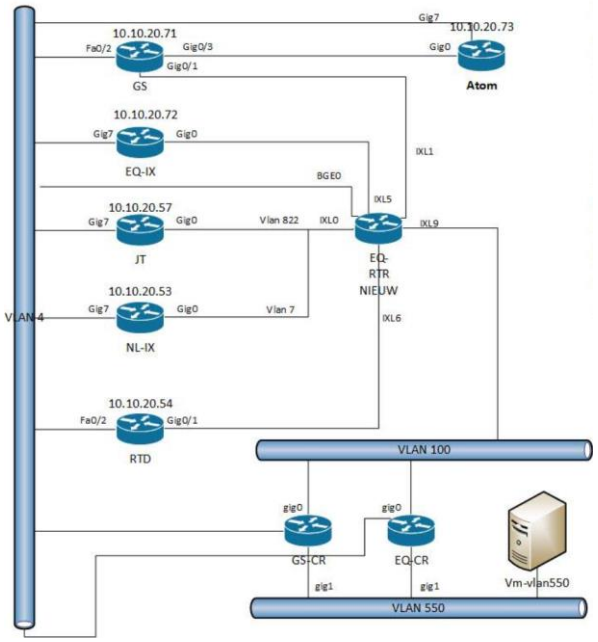
- Using x86 servers → HP Gen-10 servers, 32 CPU's and 256G memory
- Operating System to run on bare-metal
- Dedicated server to fulfill the service of full-table internet routing and routing security
- Operating System to be used: OpenBSD and plan B
- Multiple Intel X710 quad cards

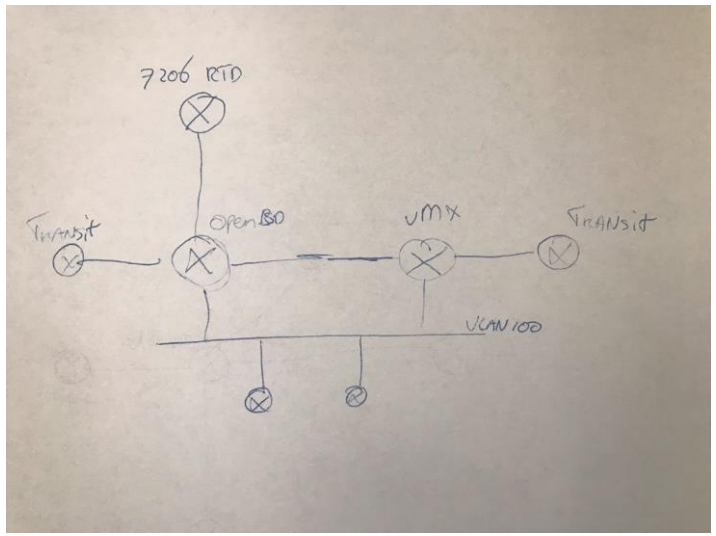
- ***Accepting that this will take time to build and test***
- ***Accepting risk of using opensource software***

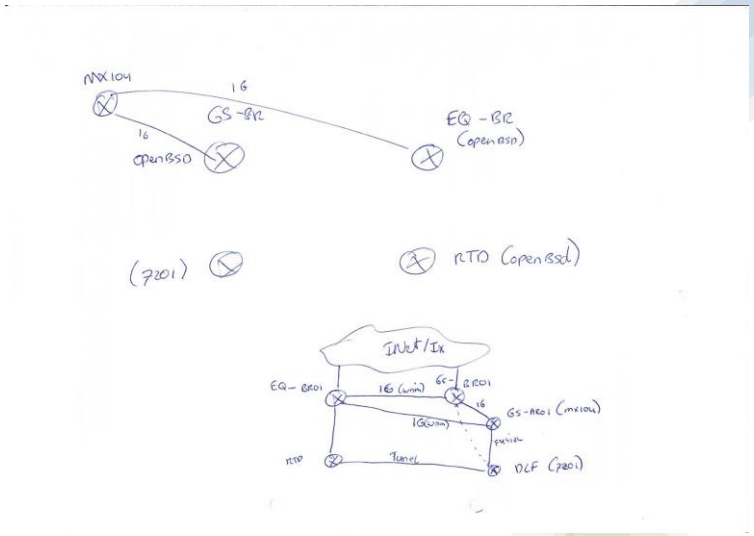
Let’s build?!

Testbed

- We created a testbed using multiple Cisco 1800, 2800 and 7200 series together with Juniper vMX and 10G switches.
- Virtual machines in Proxmox to create and verify flows/capturing traffic
- *This was not a full lab representing the production environment.*







Goal of testing

- ***Does OpenBSD meet our expectations/requirements?***

- Familiarize with the OS, reading documentation and public presentations
- Verify if all hardware is detected and working
- Verify in- and outbound routing policy and RPKI ROV
- Verification of the packet filters, flow testing
- Operability testing between other vendors for OSPF, LDP and MP-BGP
- Traffic/throughput testing

- *~100 testcases were defined and tested*

Conversions

- Interfaces
- Routing services
- Routing policy
- IP-filters

Interface conversion

- Interface group renamed from default egress→ dfz (default free zone)
  - Group name is used with packet-filtering (pf)

```
nl-ams-eq-br01:/etc$ ifconfig vlan7
vlan7: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        lladdr 40:a6:b7:51:bf:10
        description: Peering: NL-IX [10G]
        index 20 priority 0 llprio 3
        groups: vlan dfz
        status: active
        inet 193.239.117.46 netmask 0xfffffc00 broadcast 193.239.119.255
        inet6 fe80::42a6:b7ff:fe51:bf10%vlan7 prefixlen 64 scopeid 0x14
        inet6 2001:7f8:13::a501:5693:1 prefixlen 64
```

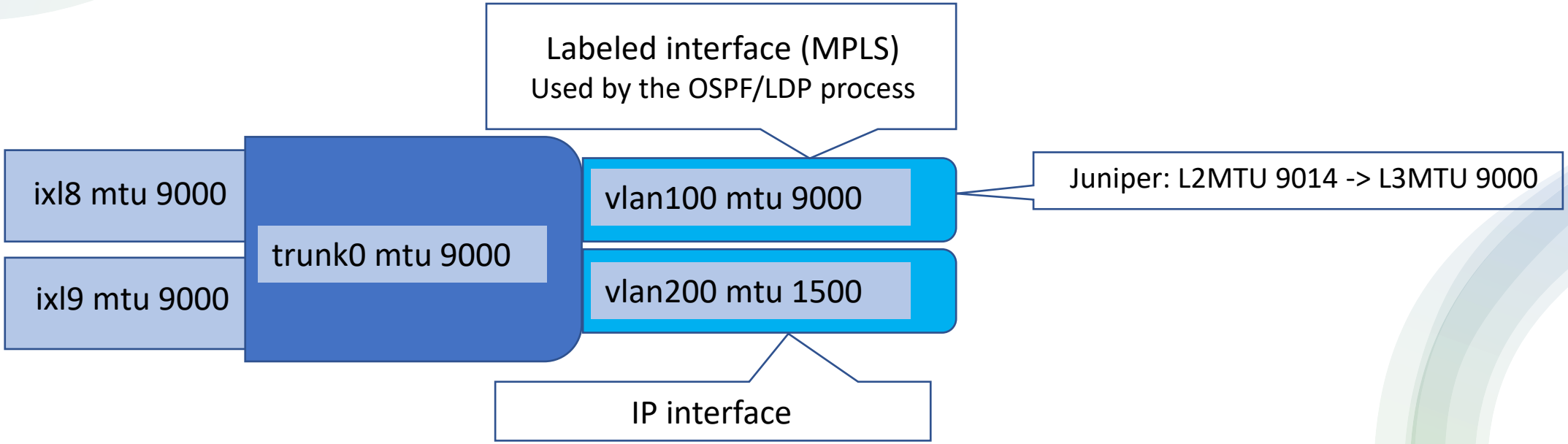
- Examples:
  - trunk0 (multiple physical interfaces as one logical), ixl8+ixl9
  - vlan100 (vlan based interface) on top of the trunk0 interface

```
nl-ams-eq-br01:~$ ifconfig trunk0
trunk0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 9000
        lladdr d4:f5:ef:13:50:b0
        description: Core: nl-ams-eq-cr01 - Po2
        index 14 priority 0 llprio 3
        trunk: trunkproto lacp
        trunk id: [(8000,d4:f5:ef:13:50:b0,4075,0000,0000),
                  (8000,01:e0:52:00:00:01,0002,0000,0000)]
                ixl9 lacp actor system pri 0x8000 mac d4:f5:ef:13:50:b0, key 0x4075, port pri 0x8000 number 0xa
                ixl9 lacp actor state activity,aggregation,sync,collecting,distributing
                ixl9 lacp partner system pri 0x8000 mac 01:e0:52:00:00:01, key 0x2, port pri 0x8000 number 0x412
                ixl9 lacp partner state activity,aggregation,sync,collecting,distributing
                ixl9 port active,collecting,distributing
                ixl8 lacp actor system pri 0x8000 mac d4:f5:ef:13:50:b0, key 0x4075, port pri 0x8000 number 0x9
                ixl8 lacp actor state activity,aggregation,sync,collecting,distributing
                ixl8 lacp partner system pri 0x8000 mac 01:e0:52:00:00:01, key 0x2, port pri 0x8000 number 0x212
                ixl8 lacp partner state activity,aggregation,sync,collecting,distributing
                ixl8 port active,collecting,distributing
        groups: trunk
        media: Ethernet autoselect
        status: active
```

```
nl-ams-eq-br01:/etc$ ifconfig vlan100
vlan100: flags=88843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST,MPLS> mtu 9000
        lladdr d4:f5:ef:13:50:b0
        description: Core: nl-ams-eq-cr01
        encap: vnetid 100 parent trunk0 txprio packet rxprio outer
```

Interface conversion - MTU stacking

- Interface MTU is also the L3MTU



Routing daemons – OSPF/LDP

- OSPF and LDP are very basic in features and configuration
- L2VPN interworking only for “ethernet”

ospf(6)d.conf

```
# macros
id="195.191.121.248"

# global configuration
router-id $id
fib-update yes
# stub router no
# spf-delay 1
# spf-holdtime 5

# auth-key secret
# auth-type simple
hello-interval 2
# metric 10
# retransmit-interval 5
router-dead-time 8
# router-priority 1
# transmit-delay 1

#redistribute default set { metric 10 type 1 }

# areas
area 0.0.0.0 {
        #gs-br01
        interface ixl1 {
                metric 10
                type p2p
        }
```

ldpd.conf

```
# global configuration
router-id 195.191.121.248
# fib-update no
# transport-preference ipv4

address-family ipv4 {
        interface ixl1
}
```

```
l2vpn vlan550 type vpls {
        bridge bridge550
        interface vlan550

        pseudowire mpw550 {
                neighbor-id 195.191.121.251
                pw-id 550
        }
}
```

OpenBGPD - Routing policy and BGP peers

- The juniper routing policy was easy to convert
- The routing policy resulted in less entries with similar functionality
- BGP groups and peers were converted using a python script
- RPKI ROV integration (with rpki-client) is enabled by removing a # from crontab

Routing policy and BGP peer examples

```
nl-ams-eq-br01:/etc$ doas bgpctl show | grep Cloud
Cloudflare-v4 #1        13335     579452     355131     0 17w4d07h    3785/20000
Cloudflare-v4 #2        13335     423776     354550     0 02w6d13h    3770/20000
Cloudflare-v4 #3        13335     396452     355131     0 17w4d07h    3763/20000
Cloudflare-v6 #1        13335     377536     355131     0 17w4d07h     380/2000
Cloudflare-v6 #2        13335     366811     354548     0 02w6d13h     376/2000
Cloudflare-v6 #3        13335     374078     355131     0 17w4d07h     359/2000
```

BGP status

```
group "nlix" {
        role peer
        set localpref 150
        neighbor 193.239.116.255 {
                role rs-client
                remote-as 34307
                enforce neighbor-as no
                descr "NL-IX RS-v4 #1"
                max-prefix 180000
        }
```

A BGP route-server neighbor with multiple “knobs”, hierarchy and ASPA roles

```
# deny RPKI invalid, built by rpki-client(8), see root crontab
deny quick from group "transit" ovs invalid
deny quick from group "nlix" ovs invalid
deny quick from group "eqix" ovs invalid
#deny ASPA invalid
deny quick from group "transit" avs invalid
deny quick from group "nlix" avs invalid
deny quick from group "eqix" avs invalid
```

Routing policy used for RPKI ROV and ASPA

```
#nexthop self for all eBGP neighbors
match from ebgp set { nexthop self }
```

Setting nhs

```
# Add internal community to transit and IX
match from group "nlix" set { community 15693:1000 }
match from ebgp AS 24785 set { community 15693:3000 }
match from group "eqix" set { community 15693:10000 }
```

Setting a community

```
#do not announce to google on eqix
#match to 2001:7f8:83::2:4115:1 set { large-community 24115:0:15169 }
#match to 2001:7f8:83::2:4115:2 set { large-community 24115:0:15169 }
```

BGP traffic engineering using large communities

```
# Add rtlabel for advertisements towards bgp pe-ce customers
match from ibgp ext-community rt 15693:3 set { rtlabel mgmt }
```

VPNv4 policy, set rtlabel to mgmt

```
nl-ams-eq-br01:~$ doas bgpctl show rib table Adj-RIB-In ovs invalid  | grep 24785
      !-? 23.133.8.0/24         217.170.19.66     100      0 24785 6939 134823 i
      !-? 23.133.8.0/24         217.170.19.65     100      0 24785 6939 134823 i
      !-? 23.139.40.0/24        217.170.19.66     100      0 24785 6939 945 60326 i
```

RIB-In, viewing RPKI invalid routes

Packet filter (pf)

- **Stateful vs stateless filtering?**

- Explicit deny, we allow what we think is needed → more on this later ☺
- Most rules implemented using ‘quick’ → predictable behaviour
- Using lists, tables and macro’s in the pf rules
- Similar type of syntax is used in the routing policy

Packet filter (pf) examples

```
table <martians> { 0.0.0.0/8 10.0.0.0/8 127.0.0.0/8 169.254.0.0/16      \
                   172.16.0.0/12 192.0.0.0/24 192.0.2.0/24 224.0.0.0/3 \
                   192.168.0.0/16 198.18.0.0/15 198.51.100.0/24        \
                   203.0.113.0/24 }

table <martiansv6> { ::/8 0100::/64 2001:2::/48 2001:db8::/32 2002::/16 \
                     fc00::/7 3ffe::/16 }
```

```
# Drop packets from non-routable addresses immediately
block in quick on dfz from <martians> to any
block in quick on dfz from <martiansv6> to any
```

```
table <bc_prefixes> { 46.183.248.0/21 195.191.120.0/23
```

```
# Drop packet from our own as space immediately (antispoofing)
block in quick on dfz from <bc_prefixes>
```

```
#Pass ipsec tunnels and pptp
pass in quick on dfz proto { ah, esp, gre } to <bc_prefixes> no state
```

```
#Allow all tcp and udp traffic towards bc prefixes
pass in quick on dfz proto { tcp, udp } to <bc_prefixes> no state
```

Combination of a list and table

```
icmp6_types="{ 1, 2, 3, 128, 129, 135, 136 }" # destination unreachable, packet too big, time exceeded, echo request, echo reply (ping6)
#icmp6_types_ext_if="{ 128, 135, 136 }"
```

Macro example

```
pass in quick on dfz inet6 proto ipv6-icmp icmp6-type $icmp6_types no state
```

```
#Allow incoming eBGP sessions on directly connected networks only
pass in quick on dfz proto tcp from self:network to self port 179 no state
```

```
# Block everything else
block in log on dfz all
block in log on oobmgmt all
```

Management and security

- Management & Routing domains
- System hardening

Management and Routing domains

- By using r(outing)domains we can create separate routing tables.
- We defined two rdomains:
  - Inband management → rdomain 3 (using MPLS L3VPN)
  - Inband management using other IP space → rdomain 4 (using static routes)

```
pass in quick on ixl2 inet proto tcp from $trusted_ip to <oob_prefix> port { 2222, 3389 } rtable 4
```

- IP interfaces are assigned to the rdomain

```
nl-ams-gs-br01:~$ netstat -T 3 -rn
Routing tables

Internet:
Destination        Gateway            Flags   Refs      Use   Mtu  Prio Iface
default            192.168.255.254    UGS        0 36076500     -     8 vlan4
127.0.0.1          127.0.0.1          UHl        0        0     -     1 mpe0
192.168.56/24      192.168.255.2      UGS        0       10     -     8 vlan4
192.168.56/24      195.191.121.248    UT         0        0     -    48 mpe0
192.168.194/24     195.191.121.253    UT         1 70618210     -    48 mpe0
192.168.214/24     195.191.121.251    UT         0 23352814     -    48 mpe0
192.168.224.1/32   195.191.121.250    UT         0        0     -    48 mpe0
192.168.254/23     192.168.255.239    UCn       17 17596803     -     4 vlan4
192.168.254/23     195.191.121.248    UT         0        0     -    48 mpe0
```

```
nl-ams-gs-br01:~$ cat /etc/hostname.vlan4
rdomain 3
rtlabel mgmt
parent trunk0 vnetid 4
inet 192.168.255.239/23
description "Management vlan"
!route -T3 -n add default 192.168.255.254 -label mgmt
#exchange
!route -T3 -n add 192.168.56.0/24 192.168.255.2 -label mgmt
group mgmt
up
```

Hardening of the system

- OpenBSD is a hardened OS and preaches to be “secure by default”
- Disabling unused default services:
  - Example: slaacd, dhcp-service (client), quotachecks, sndiod disabled
- Services for management → isolated in rdomains
- Internet facing services
- Infrastructure IP space

} Packet filter

Migrations and upgrades

- Migration attempt highlights
- A migration special
- Caveats after the migration
- Our method of upgrading the system

Migration attempt #1

- Preparation:
  - Patching scheme for the physical cabling
  - Migration order and verification documentation

- We replaced one border router with OpenBSD and ran into several issues

- ***We were not in control***, so we did a rollback and evaluation.

Migration attempt #2

- Fixed issues from attempt #1 in the lab
- Added troubleshooting of pf rules and flows

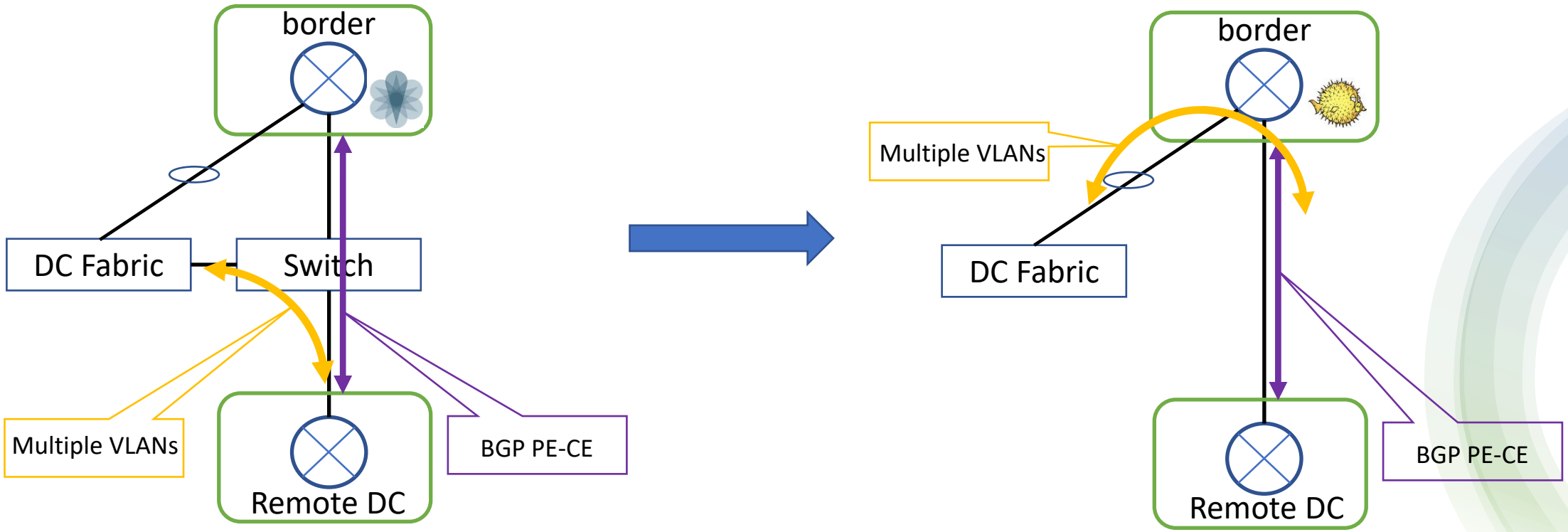
```
#troubleshooting flows
#tshootip1="{ 47.240.90.48/32 }"
#tshootip2="{ 2a01:8800::25/128 }"
#pass in log quick from $tshootip1 no state
#pass in log quick proto tcp from $tshootip2 no state
```

- Created documentation on how to receive more logging from network services
- De-peering of IPv6

- ***We were in control!*** ☺

Specials…. and 2nd border router migration

- vlan naming/numbering scheme: ixl/trunk number + vlan-id = name of the vlan interface.
  - vlan 4 on ixl8: vlan84, vlan 4 on trunk0: vlan4, veb interface with vlan 4: veb4
  - veb4 has vlan84+vlan4 added as vports
  - One veb was routing within a rdomain for MGMT
  - The amount of interfaces adds up quickly >50



Caveats >1 month

- In OpenBSD 6.9 we could not restart ldpd when bgpd was running with full-tables
- ospf6d network type on interfaces acts unpredictable → best results with p2p
- We encountered a bug in the route-reflection of VPNv4
- Reachability issues towards some RPKI repositories

```
nl-ams-eq-br01:/etc$ doas route sourceaddr
Preferred source address set for rdomain 0
IPv4: 195.191.121.248
IPv6: 2a01:8800:0:1::1
nl-ams-eq-br01:/etc$
```

- Using “doas” on linux machines ☺

Upgrading the system

- We started with OpenBSD 6.9 stable, new stable is released every 6 months, latest: OpenBSD 7.4
- x86 server and components
- Downgrading
- Countermeasures
- Backup scripts
- Maximum of 1 year behind stable releases

Implemented features and future

- ✅ Implementation of Large community support
- ✅ Implementation of Multipath routing
- ✅ Implementation of BGP Add-path
- ✅ Implementation of ASPA validation
- FlowSpec integration in OpenBGPD → API available in 7.4

- ✅ Sponsoring a public OpenBSD mirror
- Keep reading and contributing to tech@ and bugs@

Conclusion

- This “journey” took us around 6 months
- Building a lab that was production-like was a good move
- Small concessions on the network design and features we used
- Border routers are stable, 4 flawless system upgrades
- We submitted about 5 bug reports, 3 of them were fixed
- Cost effective solution for our requirements
- Exposure to some latest routing security developments like ASPA

- **Was it all worth it? → YES**

Special thanks:

My employer Routz for sponsoring this day!

https://routz.nl

Eddy Mouws – CTO

BUSINESS CONNECT

https://businessconnect.nl