Towards an IP fabric topology @ Skroutz

...an early production prototype

Alexandros Afentoulis, Stefanos Boglou
June 12th 2019, GRNOG 8, Tower of Books, SNFCC

What is `skroutz` ?

Price search/aggregator for consumer products

3440 affiliated shops

750.000 unique visitors per day

10.000.000 page views per day

About 400mbit/s average front-end traffic (HTTPS)

Existing network stack (Main Site)

- 2 core/edge IP routers running Debian (see apoikos' pres at GRNOG 2)
- A virtual-chassis, 4 Juniper EX4200 stacked
  - SPOF (almost), control plane shared among line cards
  - Stiff to maintain/upgrade
  - Limited scaling/expanding capabilities
  - Vendor lock-in
- Buffers issues on switches, potentially leading to packet drops
- Stack members ports already full
- Increased need for east-west traffic capacity

Next generation requirements

- Focus on the switches stack
- Maintainable infrastructure that scales
- Increase fault-tolerance
  - Reduce failure-domains, minimize broadcast domains
  - Fast convergence in case of failure
- Avoid vendor lock-in and proprietary tech limitations
- High link utilization
- Avoid overlay network complexity if possible
- Linux hosts integration

IP Fabric!

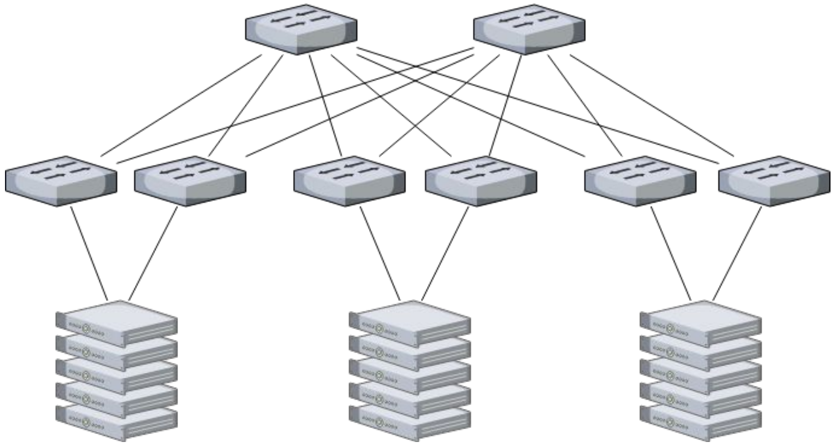
- Hot trend/topic for data-center networking
- Promises for scalability and flexibility
- A couple of “known” implementations out there
- RFCs backing specific choices, e.g. RFC7938 for BGP
- Lots of choices regarding vendors, protocols, topologies

How does it fit

- Limits the scope of failure domains
  - Broadcast domains with up to 2 devices
  - Each device has its own control plane (eBGP)
- eBGP features
  - Standards-compliant across vendors
  - Fast convergence on failures (with tuned timers and BFD)
  - Traffic engineering, eg drain device traffic, Load-Balance Layer7 load-balancers
  - ECMP (Equal Cost MultiPath) ➔ Load-Balance links (replaces LACP)
- Scalable architecture
- Debian hosts can join IP Fabric as an additional tier

Our implementation ingredients

- Leaf-spine topology
  - Two leaf switches per rack
  - Two spine switches
  - Juniper QFX5100{48S,24Q}
- IP data plane
- eBGP control plane
- AS and IP numbering scheme
- Ansible and Puppet
- Bird routing daemon on Debian



First Iteration

- Gains expected from first iteration:
  - Production level implementation
  - Prepare Ansible and Puppet to ease/automate deployment
  - Evaluate better monitoring solutions to munin/SNMP
  - Familiarize our team with basic IP fabric concepts
  - Alleviate a big load from the current switch stack
  - Simulate failures on a non-critical production network
- Hardware:
  - 2 leaf switches (no spines at this point)
  - 8 (production) Debian ganeti nodes
- VMs disk replication & memory transfer over IP fabric

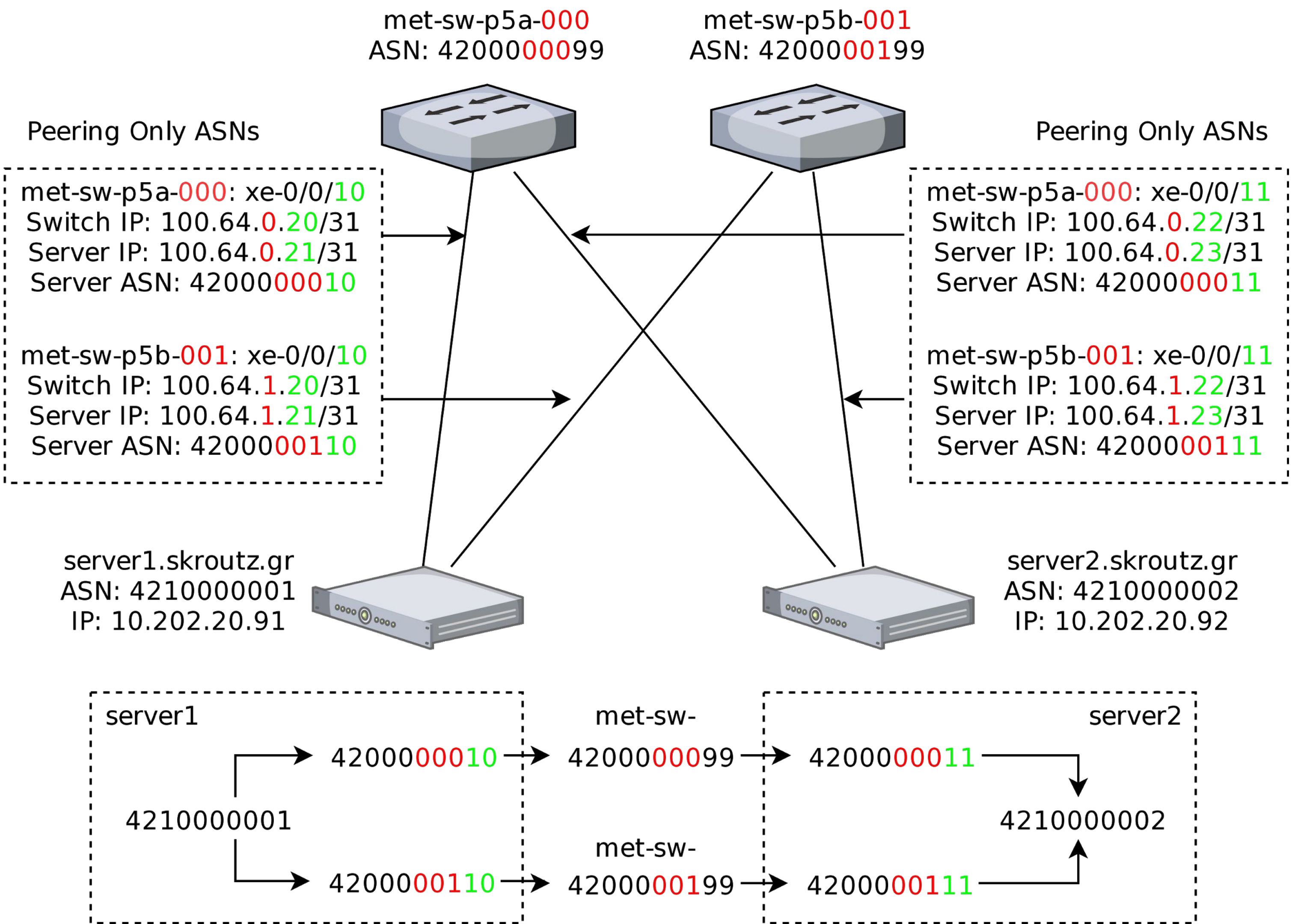
AS & IP numbering

- Algorithm/scheme to predictably devise AS numbers and IP ranges
- No (need for) coordination between Ansible and Puppet
- Coupling configuration for switches & servers
- Pre-configure all eBPG peerings on switches' side

- Hijack CGNAT space for IP Fabric peerings, 100.64.0.0/10
- 32bit private AS numbers, 42000xxxyy

Numbering walkthrough

- For each leaf switch we assign:
  - An 3-digit integer **xxx** encoded in hostname, eg. met-sw-p5b-**001**
  - A 100.64.xxx.0/24 IP range for peerings, e.g 100.64.1.0/24
  - A hundred private AS numbers, like AS42000xxxyy, e.g. AS42000**001**{00-99}
- ASN distribution
  - Leaf switch gets the last ASN, peers the rest based on peering iface
  - e. g. switch local-as 42000001**99**, xe-0/0/7 peer-as 42000001**07**
- IPs distribution
  - a /31 for each p2p link, switch gets the even, peer gets the odd
  - e.g. xe-0/0/14: 100.64.**1**.28/31, peer IP 100.64.**1**.29/31, peer AS 42000**001**14



met-sw-p5a-000
ASN: 4200000099
met-sw-p5b-001
ASN: 4200000199
Peering Only ASNs
met-sw-p5a-000: xe-0/0/10
Switch IP: 100.64.0.20/31
Server IP: 100.64.0.21/31
Server ASN: 4200000010
met-sw-p5b-001: xe-0/0/10
Switch IP: 100.64.1.20/31
Server IP: 100.64.1.21/31
Server ASN: 4200000110
Peering Only ASNs
met-sw-p5a-000: xe-0/0/11
Switch IP: 100.64.0.22/31
Server IP: 100.64.0.23/31
Server ASN: 4200000011
met-sw-p5b-001: xe-0/0/11
Switch IP: 100.64.1.22/31
Server IP: 100.64.1.23/31
Server ASN: 4200000111
server1.skroutz.gr
ASN: 4210000001
IP: 10.202.20.91
server2.skroutz.gr
ASN: 4210000002
IP: 10.202.20.92
server1
4210000001
4200000010
4200000110
met-sw-
4200000099
met-sw-
4200000199
server2
4200000011
4200000111
4210000002

Ansible on switches

- Pre-configure as much as possible
- Juniper’s ansible role
- Home-grown roles for IP fabric switches
- Custom lookup plugin implementing the addressing algorithm
- Configure:
    - Virtual-router routing instance with eBGP protocol
    - Separate BGP group for servers (neighbor ASNs, IPs, import/export, BFD)
    - import/export policies filtering network prefixes
    - Analytics, push interfaces/queues stats for graphing

Puppet

- Fetch switches' hostname and port from LLDP (layer2!)
  - LLDP => layer2 protocol, no configuration needed
- Custom puppet function transcodes LLDP facts to IPs and ASNs
- Configures /etc/network/interfaces (debian-based only)
  - Custom `iface` resource for managing network interfaces
  - "Peering" interfaces with switches
  - Dummy interface with /32 (/128) addresses to announce
- Configures eBGP on bird
  - Bird is our eBGP/routing daemon of choice
  - Control plane that listens and announces layer3 IPs to and from IPFabric

Routing on Debian hosts

```
bird> show route for 10.202.20.93 all
10.202.20.93/32  via 100.64.0.28 on eth5
[met_sw_p5a_000 11:44:41] * (100) [AS4210000003i]
    Type: BGP unicast univ
    BGP.origin: IGP
    BGP.as_path: 4200000099 4200000004 4210000003
    BGP.next_hop: 100.64.0.28
    BGP.local_pref: 100
                via 100.64.1.28 on eth4
[met_sw_p5b_001 11:44:41] (100) [AS4210000003i]
    Type: BGP unicast univ
    BGP.origin: IGP
    BGP.as_path: 4200000199 4200000104 4210000003
    BGP.next_hop: 100.64.1.28
    BGP.local_pref: 100
```

```
root@hp1.gnt-:~# ip r
default via 185.6.77.33 dev bond0 onlink
10.42.2.0/24 via 10.202.20.1 dev replication
10.202.20.0/24 dev … src 10.202.20.91
10.202.20.92 proto bird src 10.202.20.91
    nexthop via 100.64.0.0  dev eth5 weight 1
    nexthop via 100.64.1.0  dev eth6 weight 1
10.202.20.93 proto bird src 10.202.20.91
    nexthop via 100.64.0.0  dev eth5 weight 1
    nexthop via 100.64.1.0  dev eth6 weight 1
10.202.20.94 proto bird src 10.202.20.91
    nexthop via 100.64.0.0  dev eth5 weight 1
    nexthop via 100.64.1.0  dev eth6 weight 1
10.202.20.95 proto bird src 10.202.20.91
    nexthop via 100.64.0.0  dev eth5 weight 1
    nexthop via 100.64.1.0  dev eth6 weight 1
```

Monitoring

- Switches
  - use Juniper Analytics for graphing
  - Junos push JSON to Logstash, 5 seconds interval
  - Monitor buffer statistics with millisecond accuracy for detecting micro-bursts
  - Use grafana as a graphing tool
- Debian hosts
  - Log route changes messages via route netlink
  - Check for multipath routes existence


IPFabric Interface Statistics
Last 30 minutes Refresh every 5s
Switch All
Port All
Throughput
met-sw-p5b-001 - Total Throughput
max avg current total
RXbps Total 2.987 Gbps 380 Mbps 1.409 Gbps 135.824 Gbps
TXbps Total 2.432 Gbps 379 Mbps 1.010 Gbps 135.417 Gbps
met-sw-p5a-000 - Total Throughput
max avg current total
RXbps Total 1.877 Gbps 396 Mbps 860 Mbps 139.763 Gbps
TXbps Total 1.895 Gbps 394 Mbps 653 Mbps 139.105 Gbps
Interfaces
RX/TX BitsPerSecond (Derived)
max avg current
met-sw-p5b-001.xe-0_0_2.rxbyte 1.574 Gbps 93 Mbps 51 Mbps
met-sw-p5a-000.xe-0_0_8.rxbyte 1.356 Gbps 97 Mbps 507 Mbps
met-sw-p5a-000.xe-0_0_12.rxbyte 638 Mbps 43 Mbps 106 Mbps
met-sw-p5b-001.xe-0_0_0.rxbyte 638 Mbps 81 Mbps 4 Mbps
met-sw-p5a-000.xe-0_0_2.rxbyte 615 Mbps 135 Mbps 115 Mbps
met-sw-p5b-001.xe-0_0_12.rxbyte 605 Mbps 8 Mbps 4 Mbps
met-sw-p5b-001.xe-0_0_4.rxbyte 529 Mbps 87 Mbps 456 Mbps
met-sw-p5b-001.xe-0_0_6.rxbyte 492 Mbps 34 Mbps 468 Mbps
met-sw-p5a-000.xe-0_0_10.rxbyte 439 Mbps 82 Mbps 119 Mbps
met-sw-p5b-001.xe-0_0_10.rxbyte 427 Mbps 25 Mbps 30 Mbps
met-sw-p5b-001.xe-0_0_8.rxbyte 395 Mbps 38 Mbps 395 Mbps
met-sw-p5a-000.xe-0_0_0.rxbyte 346 Mbps 6 Mbps 3 Mbps
met-sw-p5a-000.xe-0_0_6.rxbyte 313 Mbps 12 Mbps 5 Mbps
met-sw-p5a-000.xe-0_0_4.rxbyte 282 Mbps 20 Mbps 3 Mbps
RX/TX PacketsPerSecond (Derived)

Next steps/challenges

- Move more bare metal hosts traffic (services) over the fabric
- Expand the fabric: introduce spines, add more leafs
- Move virtual machines traffic over the fabric, i.e. routing on the host
- Establish connectivity to the rest of the world (distribute default gateways?)
- Improve visibility (monitoring) over the fabric
- Address the bootstrapping step (DHCP or ?)

Thanks!