The modern, interoperable DC

Part 1: Solving “last mile” problems with BGP

Jerzy Kaczmarski
Adam Kułagowski

team@codilime.com

Introduction

10
Years in business

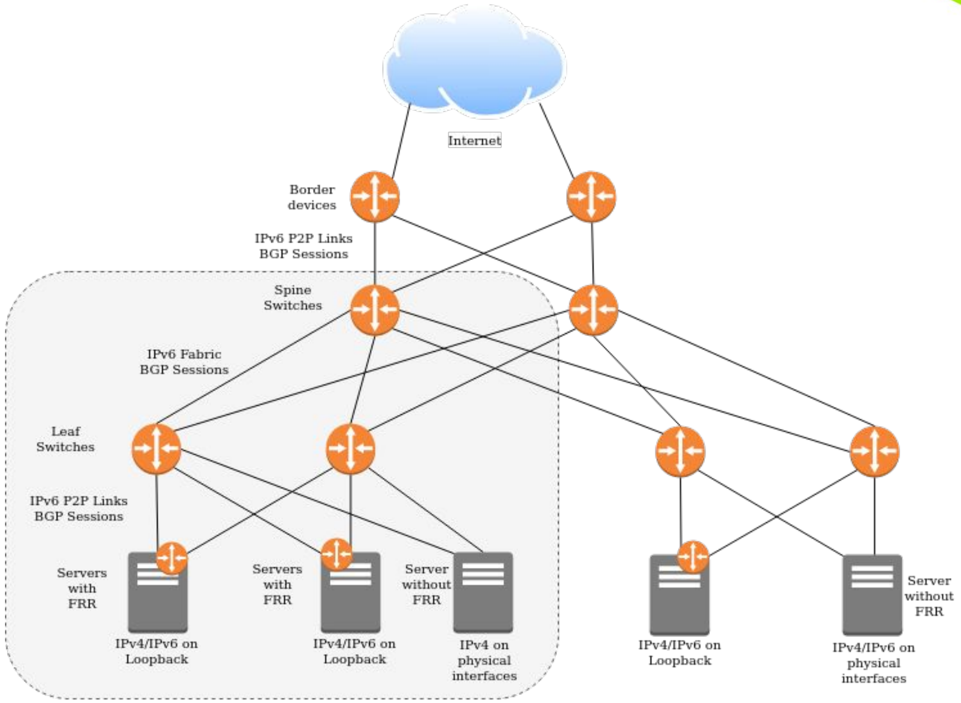
3
Offices

200+
Network, software & DevOps engineers

6
Our clients' time zones

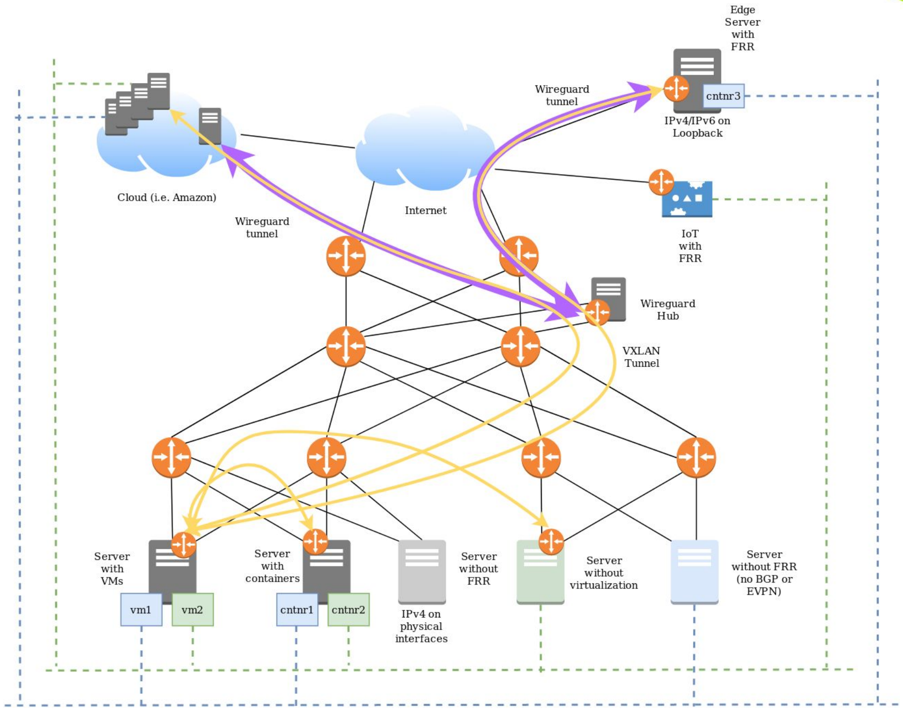
Modern DC Series - Introduction and agenda

- Scalable and easily automated networking architecture for DC environments
- Using IPv6 to simplify deployment and address management
- Using BGP for scalability, flexibility, load balancing and fast failover
- Automatically installing and configuring FRR
- Demo of working solution
- All configurations from demo will be available on github



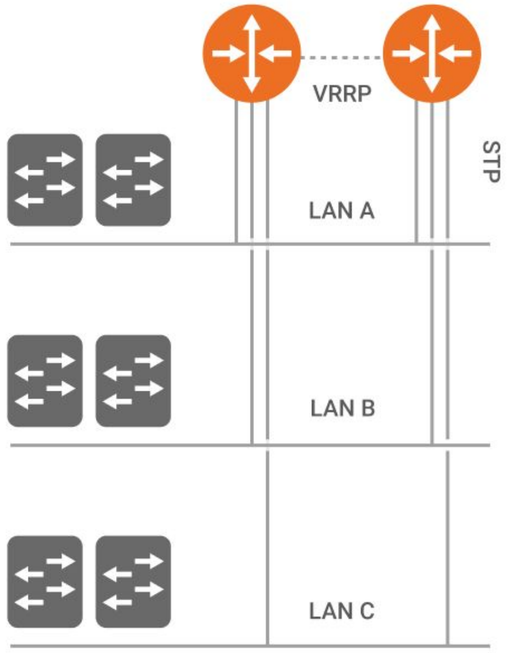
Modern DC Series - Next webinars

- Providing Layer 2 connectivity over IPv6 fabric with VXLAN and EVPN
- Multi-tenancy using virtualization, VRFs and tunnels
- Interconnecting heterogeneous resources at Layer 2:
  - legacy server without FRR or virtualization
  - server with FRR installed
  - containers (i.e. kubernetes)
  - virtual machines
- Gateways between physical and overlay networks
- Connecting two or more DCs together
- Extending Layer 2 connectivity to public clouds
- Secure tunnels for Edge Computing and IoT devices



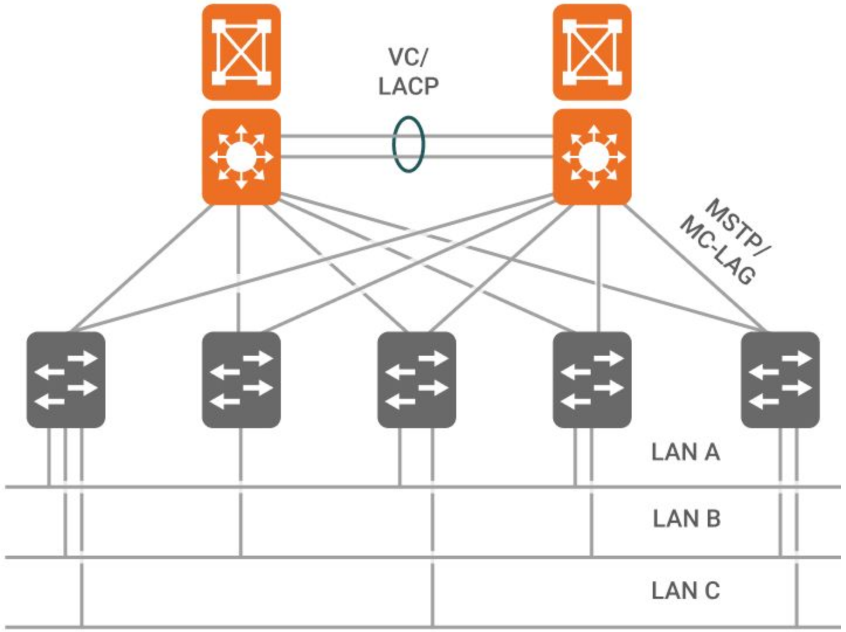
The evolution of the data center

- Pure L3 routers
- Each Network consumes one router interface
- Due to limited number of ports, Ring topology was used to connected switches within one network
- Active/standby design
- Protocols used: STP, EIGRP/OSPF, VLAN, HSRP/VRRP



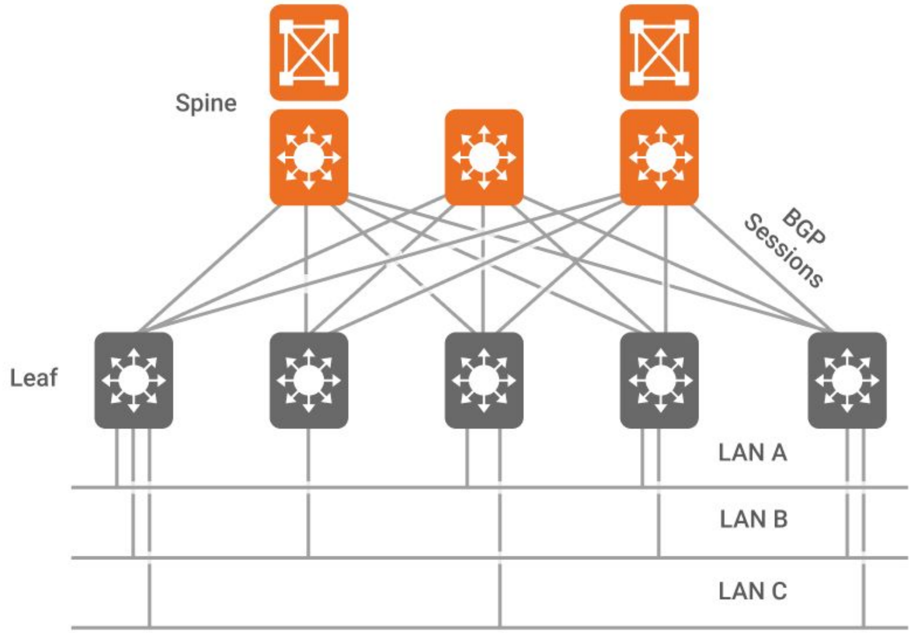
The evolution of Data Center

- L2/L3 routers
- Double-star topology
- Trunk between core & access switches
- Active/standby design
    - unless proprietary solution (such as Virtual Chassis) was used
- Vertical scalability (faster uplinks)
- Routing done on core switches
- Servers plugged into one access switch (w/ exc exception of MC-LAG)
- Protocols used: MSTP, EIGRP/OSPF, VLAN, HSRP/VRRP, LACP, VTP/GVRP, VSS



The evolution of Data Center

- L2/L3 routers
- Pure L3 traffic between network devices
- Active/Active design
- Horizontal scalability (by adding spines uplinks)
- Routing done on spines or on leafs
- Servers plugged into 2 leafs for redundancy
- Protocols used: BGP, BFD, VLAN, LACP, VxLAN



Common issues

“Last mile” problems

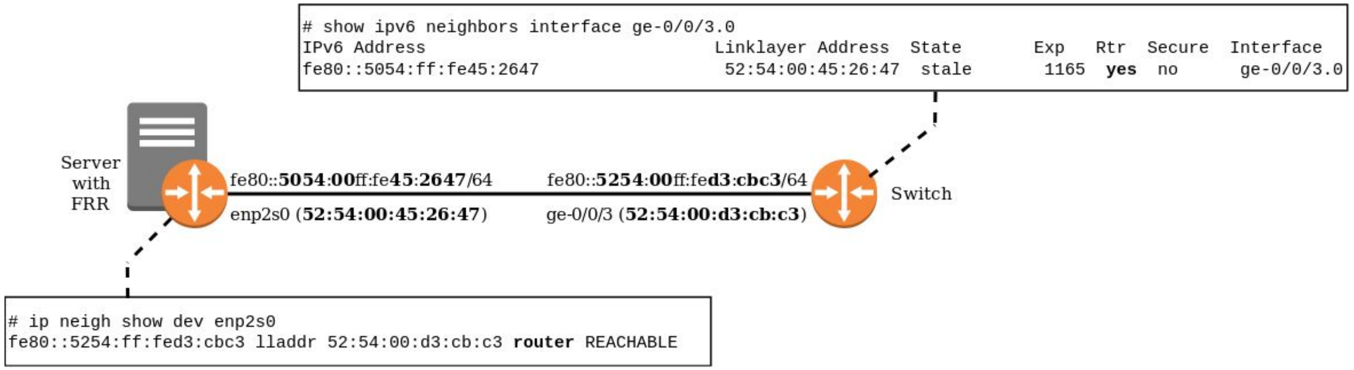
- Strong separation between servers and network (department silos)
- Complicated configuration on Leafs (VLANS, BGP policies, VXLAN configuration w/ VTEPS, multiple VRFs)
- Slow failover on servers (~3 sec) due to LACP limitation
- No Active/Active uplinks w/o EVPN

Other problems

- Layer 2 (with its problem) still present on Network equipment
- IPv4 (management) between switches/towards the servers (lots of IP space wasted for network/broadcast on /30 links)
- Service Load Balancing/Redundancy often requires external equipment/software

IPv6 Link-local and Neighbor Discovery

- Every IPv6-enabled interface is **automatically** assigned a link-local address
  - Address is often based on MAC address (EUI-64 standard)
  - Same address can be reused on different interfaces of a single device
  - No need to manually configure addresses on point-to-point links
  - Routing protocol can use link-local addresses to exchange information
- Devices configured as routers can use Neighbor Discovery Protocol (NDP) to advertise their addresses



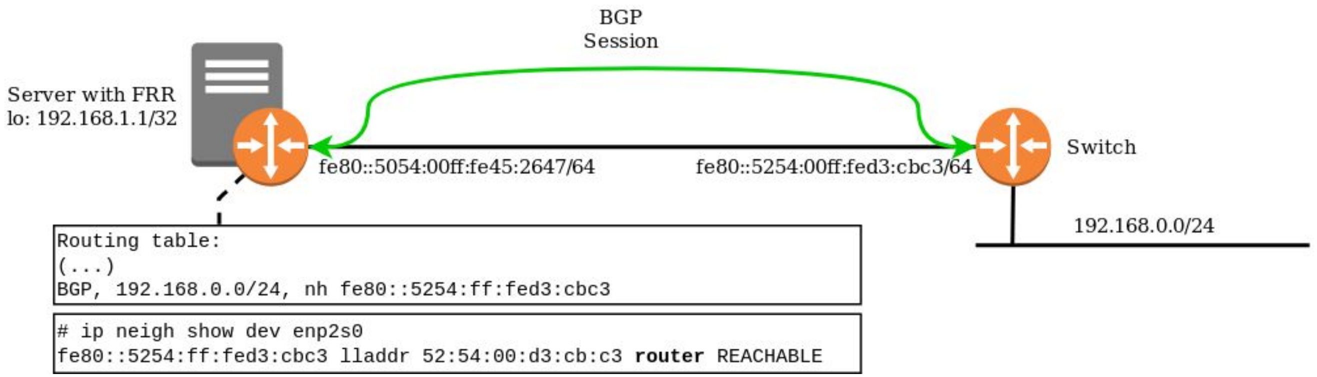
```
# show ipv6 neighbors interface ge-0/0/3.0
IPv6 Address                          Linklayer Address  State       Exp   Rtr   Secure   Interface
fe80::5054:ff:fe45:2647                52:54:00:45:26:47  stale      1165  yes   no        ge-0/0/3.0
```

```
# ip neigh show dev enp2s0
fe80::5254:ff:fed3:cbc3 lladdr 52:54:00:d3:cb:c3 router REACHABLE
```

Server with FRR
fe80::5054:00ff:fe45:2647/64
enp2s0 (52:54:00:45:26:47)
fe80::5254:00ff:fed3:cbc3/64
ge-0/0/3 (52:54:00:d3:cb:c3)
Switch

Unnumbered BGP and autodiscovery

- BGP session automatically established using link-local addresses
  - No need to set neighbor's IP or AS number in router's configuration
  - On some vendors this functionality is not built-in (requires extra scripts)
- RFC5549 - extension to BGP protocol
  - Allows IPv4 prefixes to be advertised with IPv6 next-hop
  - Possible for network core to be IPv6-only while services/applications/users use IPv4

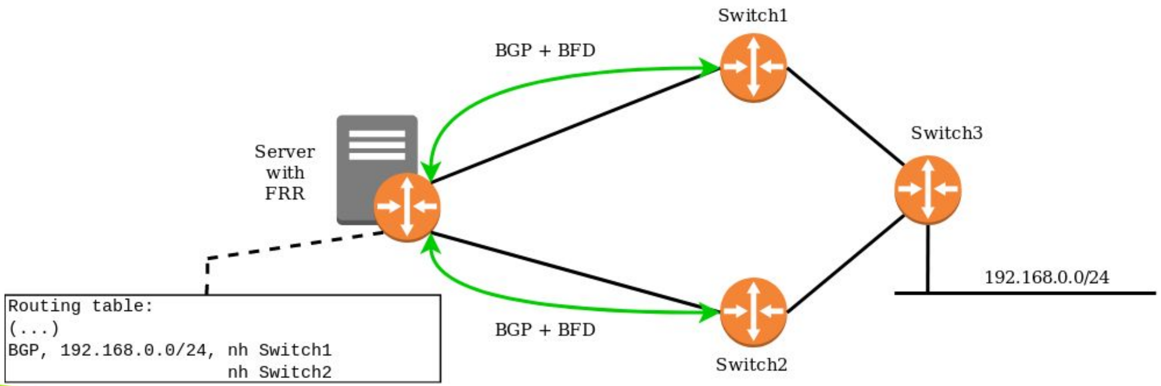


```
Routing table:
(...)
BGP, 192.168.0.0/24, nh fe80::5254:ff:fed3:cbc3
```

```
# ip neigh show dev enp2s0
fe80::5254:ff:fed3:cbc3 lladdr 52:54:00:d3:cb:c3 router REACHABLE
```

Load balancing and rapid failover

- Equal Cost Multi-Path (ECMP)
  - Balance connections between available paths to destination
  - Balancing based on hash of packet headers (i.e. srcIP, dstIP, protocol, srcPort, dstPort)
- Default BGP timers - 30s keepalive, 90s failover
- Bidirectional Forwarding Detection (BFD)
  - Simple protocol for detecting if remote device is reachable
  - Can be leveraged by routing protocols for very fast failover
  - Negotiated keepalive intervals (possible values way below 1 second)



```
Routing table:
(...)
BGP, 192.168.0.0/24, nh Switch1
                     nh Switch2
```

Free Range Routing (FRR)

Free Range Routing (FRRouting or FRR) is network routing software that supports (among others)
OSPF, BGP, IS-IS, LDP, PIM, Babel, BFD

FRRouting grew out of the free routing software Quagga, as the pace of Quagga development was frustrating some developers.

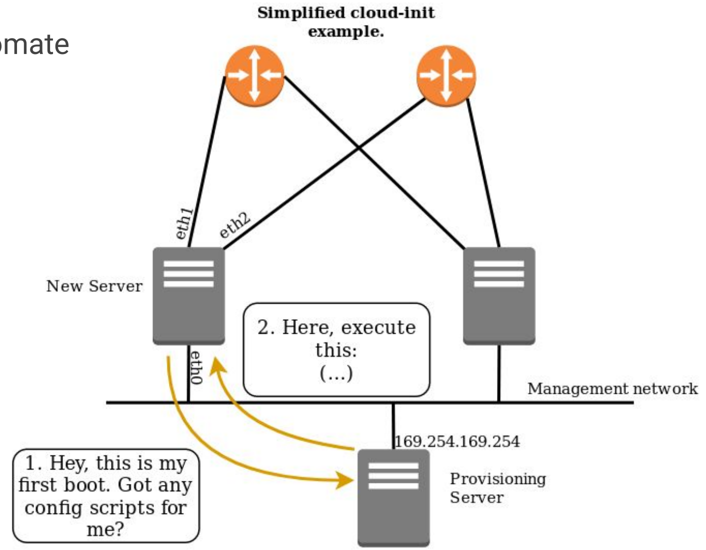
FRR contributors include Cumulus Networks (Nvidia), 6Wind and BigSwitch Networks

FRR is also a collaborative project of the Linux Foundation

Server provisioning

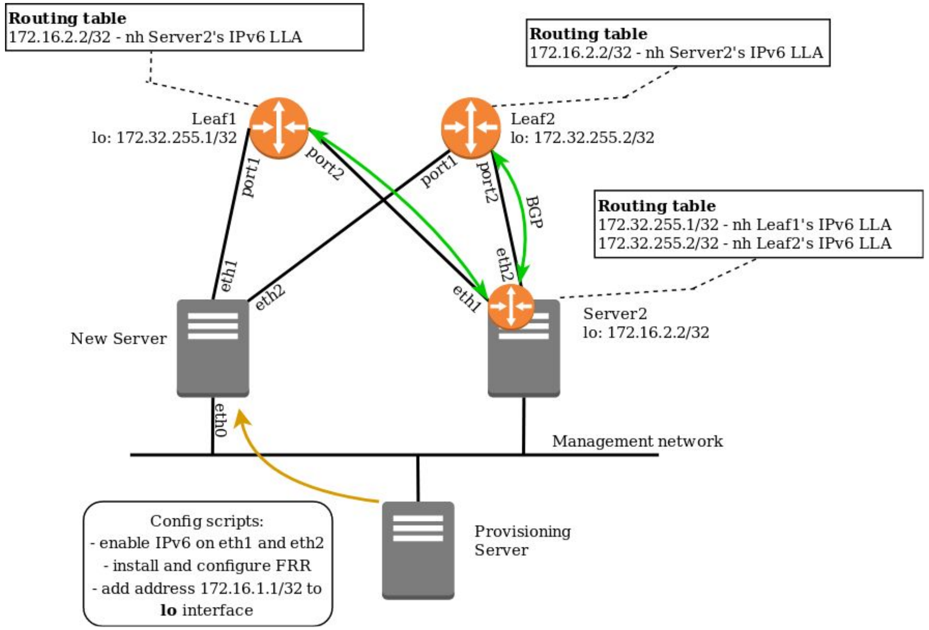
- Various tools enable operating system installation or a first boot to be customised (for example Razor, Cobbler, FAI, RackHD, MaaS, etc.) over management network
- FRR can be automatically installed and configured
- A typical FRR configuration template the same for all servers
  - The only difference is the server's IP address - one line, easy to automate

```
# cat /etc/frr/frr.conf
[...]
service integrated-vtysh-config
!
interface lo
 ip address 172.16.1.10/32
!
router bgp 64512
 no bgp ebgp-requires-policy
[...]
```


Simplified cloud-init example.
eth1
eth2
New Server
eth0
2. Here, execute this: (...)
Management network
169.254.169.254
1. Hey, this is my first boot. Got any config scripts for me?
Provisioning Server

Putting the solution together (1/2)

- Example topology:
  - Leaf1, Leaf2 and Server2 already configured
  - BGP sessions established and
  - advertising loopback addresses
  - A new server is added and connected to network
- The New Server starts:
  - Operating system is installed
  - Configuration scripts received and executed
  - FRR installed and configured

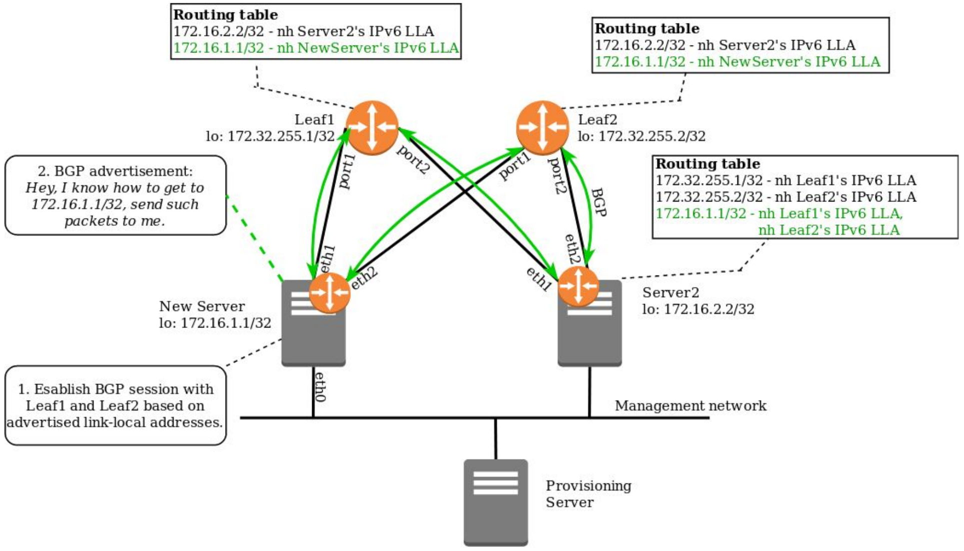


Routing table
172.16.2.2/32 - nh Server2's IPv6 LLA
Routing table
172.16.2.2/32 - nh Server2's IPv6 LLA
Leaf1
lo: 172.32.255.1/32
Leaf2
lo: 172.32.255.2/32
port1
port2
port1
port2
BGP
Routing table
172.32.255.1/32 - nh Leaf1's IPv6 LLA
172.32.255.2/32 - nh Leaf2's IPv6 LLA
eth1
eth2
eth2
eth1
New Server
Server2
lo: 172.16.2.2/32
eth0
Management network
Config scripts:
- enable IPv6 on eth1 and eth2
- install and configure FRR
- add address 172.16.1.1/32 to **lo** interface
Provisioning Server

codilime CREATING VALUE

Putting the solution together (2/2)

- The New Server sends and receives IPv6 Router Advertisements/Solicitations
- BGP sessions established to Leaf1 and Leaf2
- New Server advertises its loopback address 172.16.1.1/32
- The 172.16.1.1/32 prefix added to Leaf1 and Leaf2 routing tables. Next, Leaf1 and Leaf2 advertise this new prefix to all their BGP neighbors (Server2 in this case).
- Server2 adds 172.16.1.1/32 prefix to its routing table (two next-hops, load balancing possible)
- The New Server also receives all routes known by Leaf1 and Leaf2
- All devices can now communicate with each other



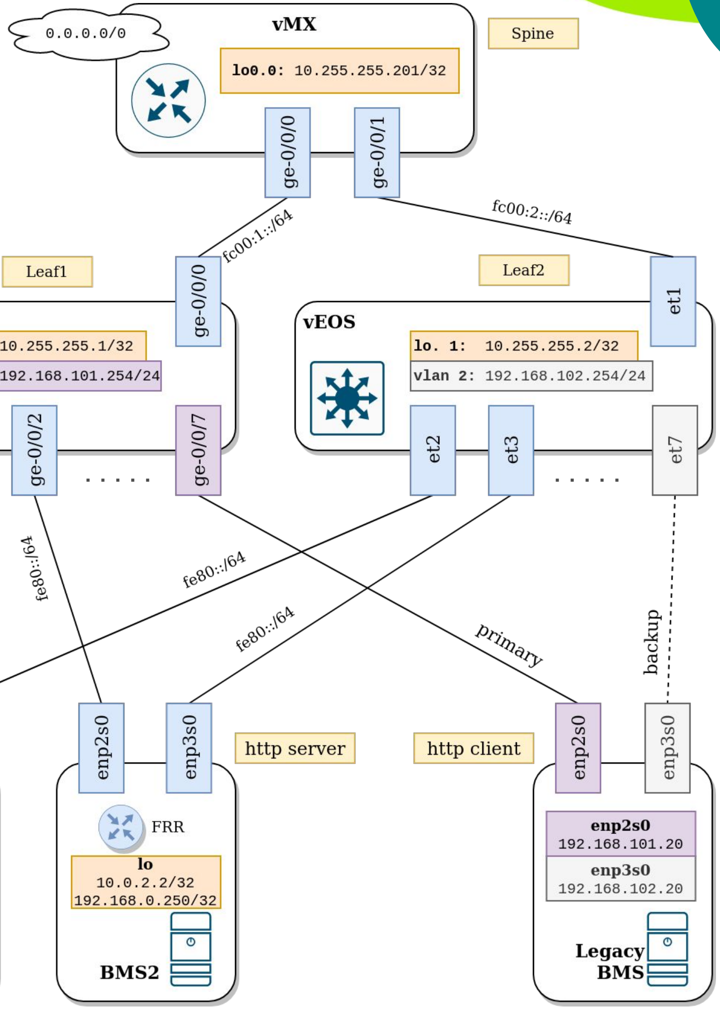
Routing table
172.16.2.2/32 - nh Server2's IPv6 LLA
172.16.1.1/32 - nh NewServer's IPv6 LLA
Routing table
172.16.2.2/32 - nh Server2's IPv6 LLA
172.16.1.1/32 - nh NewServer's IPv6 LLA
Leaf1
lo: 172.32.255.1/32
Leaf2
lo: 172.32.255.2/32
2. BGP advertisement: Hey, I know how to get to 172.16.1.1/32, send such packets to me.
Routing table
172.32.255.1/32 - nh Leaf1's IPv6 LLA
172.32.255.2/32 - nh Leaf2's IPv6 LLA
172.16.1.1/32 - nh Leaf1's IPv6 LLA, nh Leaf2's IPv6 LLA
port1
port2
port1
port2
BGP
eth1
eth2
eth1
eth2
New Server
lo: 172.16.1.1/32
Server2
lo: 172.16.2.2/32
1. Esablish BGP session with Leaf1 and Leaf2 based on advertised link-local addresses.
eth0
Management network
Provisioning Server

Demo

Demo agenda

- Topology presentation
- Launching BMS and BGP autodiscovery
- Traffic between BMS1/2 (ICMP/TCP) and ECMP
- Access from legacy BMS
- Fast failover
- Native A/A IP load balancing

Demo



codilime CREATING VALUE

Pros & cons of the proposed solution

Pros

- faster change management - removes network/server silos
- sub-second link failover
- Multiple servers uplinks in A/A without EVPN
- active/active & active/passive service failover (anycast+MED)
- pure L3 data center: no STP, no LACP, L2 issues
- easier IPv4 management
- Ability to ruch EVPN directly on servers
- no VXLAN license needed on switches
- Much simpler network configuration (no VLANS, no VRF, simpler BGP)
- Backward compatible to traditional Spine&Leaf architecture
- IPv6 ready
- Future: Flowspec

Pros & cons of the proposed solution

Cons

- More complicated solution
- Extra component on servers - FRR
- Sysadmins should know BGP basics
- Limited trust between BGP neighbors (BGP policies can protect against misconfiguration)
- New linux kernel recommended
- Quite new approach - no RFC for BGP autodiscovery
- Bugs encountered (reported to proper mainterners)

codilime® CREATING VALUE

Problems encountered

- eBPF was needed to fix traffic in one scenario
- Route table “pollution“
- No native support for BGP discovery on QFX
- Two IPV6 nexthops in IPV4 NLRI were rejected by one of the vendors.

Problems encountered - eBPF

Whenever IPv4 traffic is exchanged between server and Juniper Control Plane (ping, snmp, etc), the return IPv4 traffic is packed into IPv6 Ethertype (which is dropped by Linux)

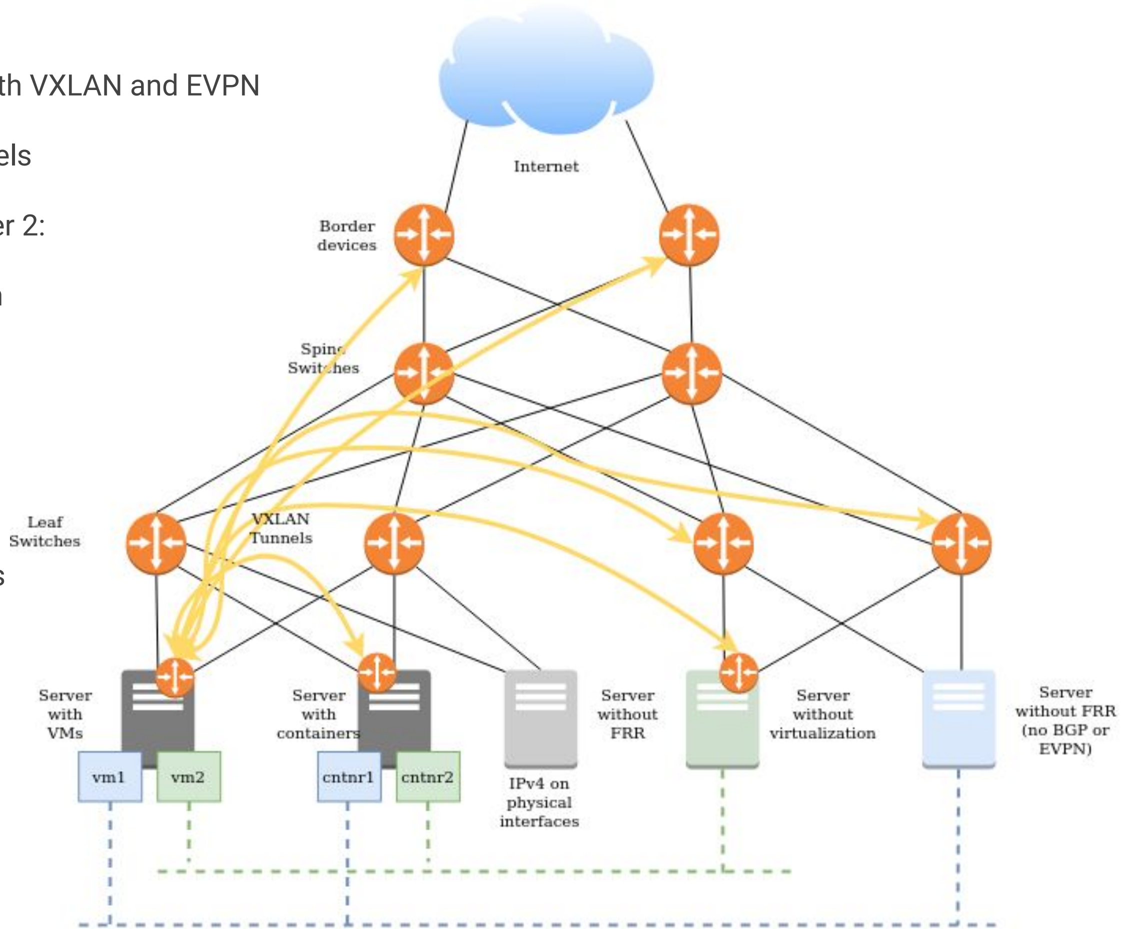
- Transit traffic and Ipv6 traffic is unaffected (medium priority)
- We should be able to work around this issue w/ the use of tc pedit/skbmod action but
  - tc lacks documentation (and existing has errors)
  - tc is after AF_PACKET so tcpdump still sees bad traffic
- At the and we applied a short bit of eBPF code (40 lines) :
  - Simple
  - Fast (can even be offloaded to NIC hardware)
  - Transparent to the rest of the Linux kernel

Summary

- The next step in the evolution of large data centers:
  - Less configuration on networking devices
  - Faster change management
  - Interoperable, open standards
  - A way to easily run services on IPv4, IPv6 or both
  - Ability to add extra features thanks to BGP
- We’re open to present the solution as a PoC or provide a FUT with a customer

Modern DC Series - Webinar 2

- Providing Layer 2 connectivity over IPv6 fabric with VXLAN and EVPN
- Multi-tenancy using virtualization, VRFs and tunnels
- Interconnecting heterogeneous resources at Layer 2:
  - legacy server without FRR or virtualization
  - server with FRR installed
  - containers (i.e. kubernetes)
  - virtual machines
- Gateways between physical and overlay networks



Questions & Answers

Configurations, source code, topology, demo recording available at:

https://github.com/codilime/modern-dc

External sources

During the demo preparation we have found the following sources/link helpful:

- O'Reilly Media “BGP in the Data Center” by Dinesh G. Dutt
- “BGP over unnumbered interfaces, automated”
  https://marcelwiget.blog/2018/06/21/bgp-over-unnumbered-interfaces-automated/

Thank you

codilime
CREATING VALUE

2100 Geng Road, Suite 210
Palo Alto, CA 94303
United States of America
+1 650 285 2458

Krancowa 5
02-493, Warsaw
Poland
+48 22 389 51 00

al. Grunwaldzka 472B (OBC)
80-309 Gdansk
Poland
+48 575 700 785

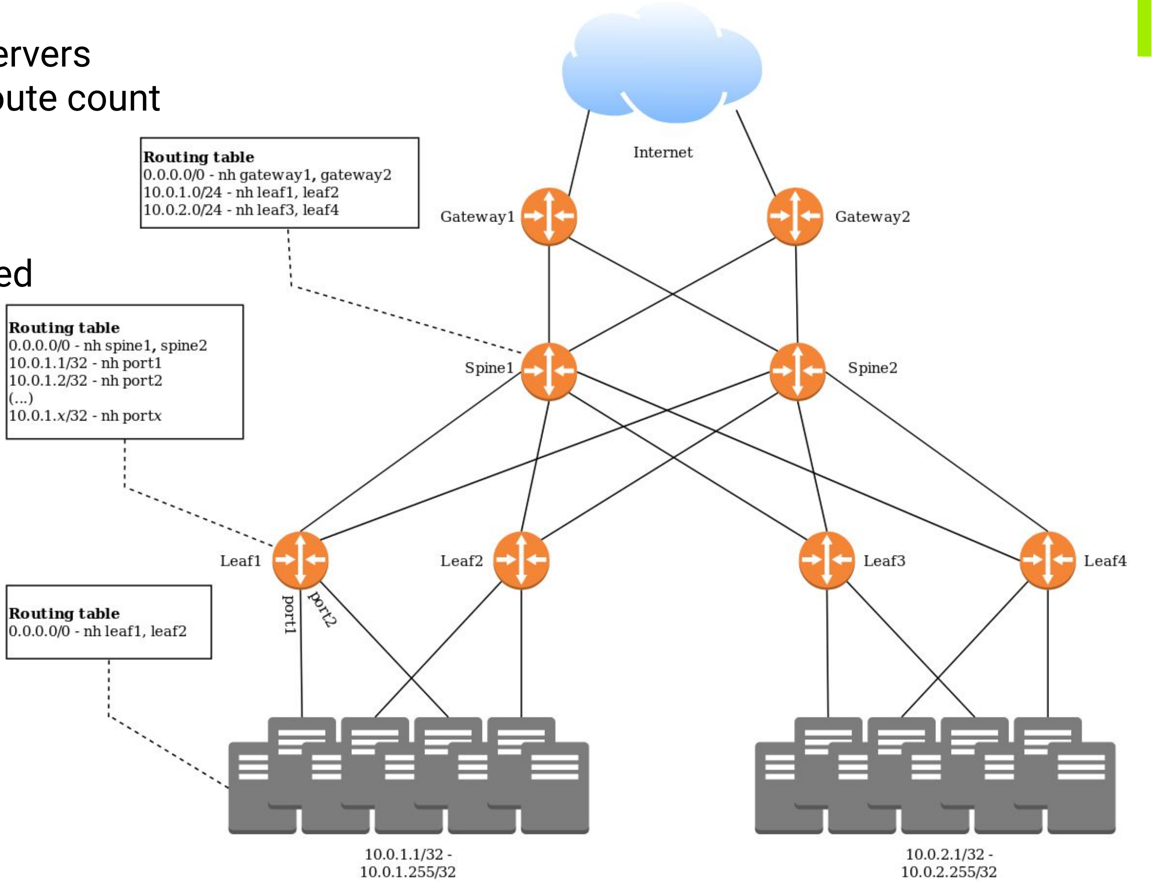
contact@codilime.com

SDN & NFV ▢ Cloud native & Multicloud ▢ Software Engineering ▢ UX / UI ▢ DevOps

Backup

Scalability - Route aggregation

- Very large DCs can have 10,000s of servers
- DCI scenarios further increase total route count
- Good planning and route aggregation significantly cut down the number of required routes
- ECMP (load balancing) can still be used
- Route filtering and aggregation done by BGP protocol
- If needed, some specific routes can be left unaggregated (i.e. for anycast services)



Routing table
0.0.0.0/0 - nh gateway1, gateway2
10.0.1.0/24 - nh leaf1, leaf2
10.0.2.0/24 - nh leaf3, leaf4

Routing table
0.0.0.0/0 - nh spine1, spine2
10.0.1.1/32 - nh port1
10.0.1.2/32 - nh port2
(...)
10.0.1.x/32 - nh portx

Routing table
0.0.0.0/0 - nh leaf1, leaf2

codilime CREATING VALUE