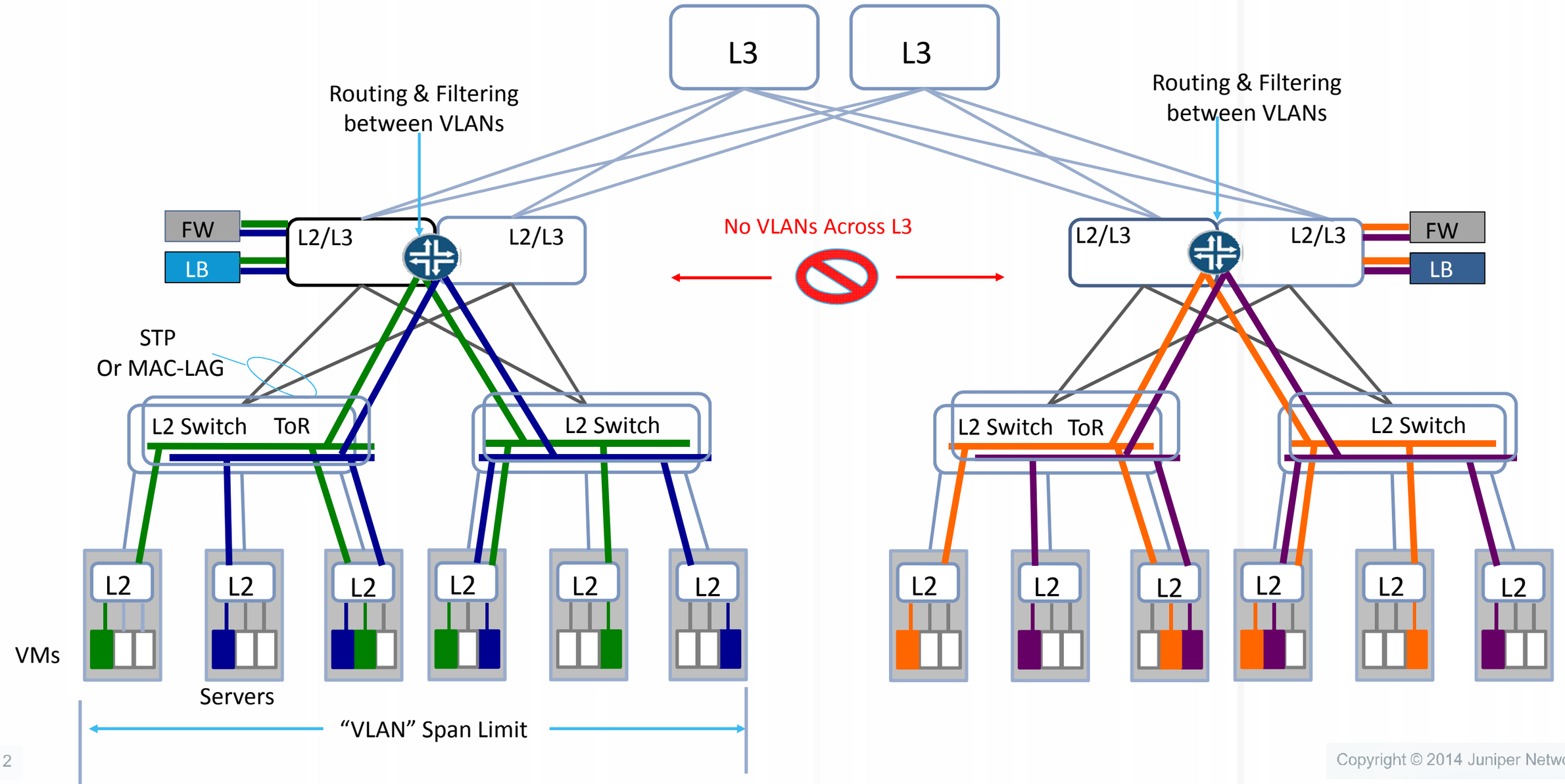
BGP dans le ToR

FRnOG 23

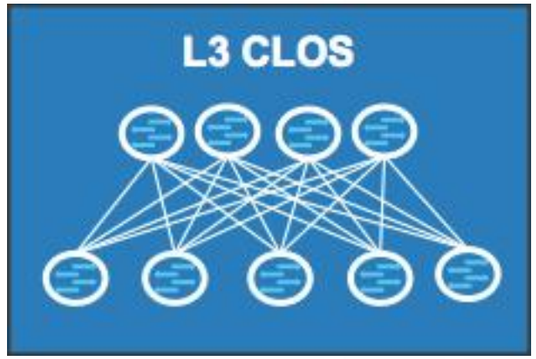
Antoine Sibout

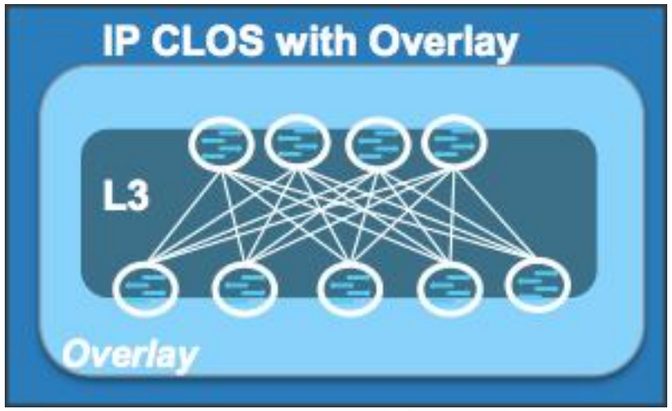
Datacenter Architect – Juniper Networks

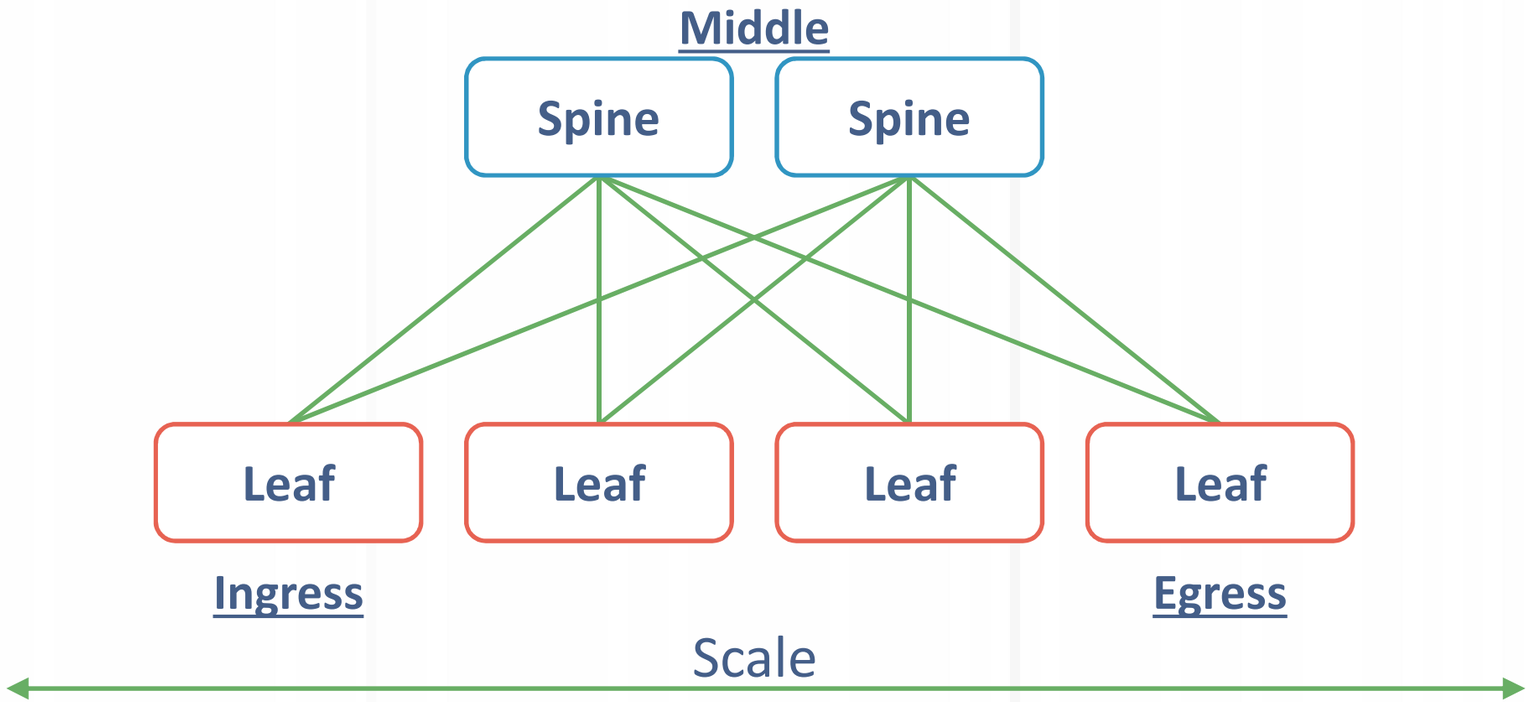
Datacenter traditionnel – “Underlay & Vlans”



Evolution vers des architectures Clos IP “Fabrics”

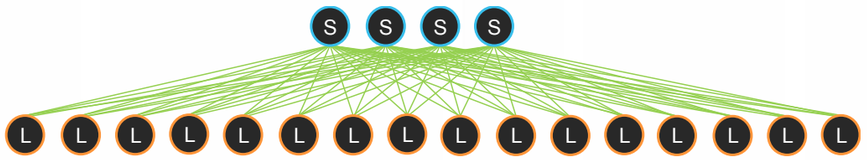




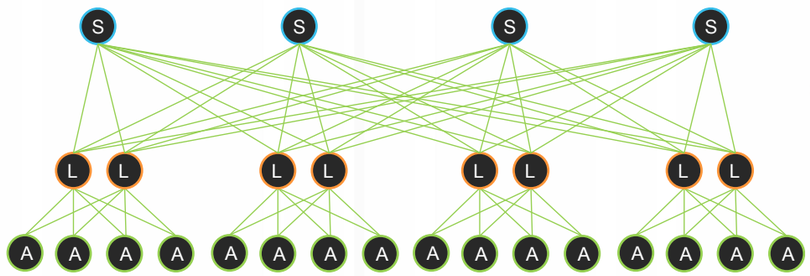


Émergence de Clos Fabrics à 3-5 étages

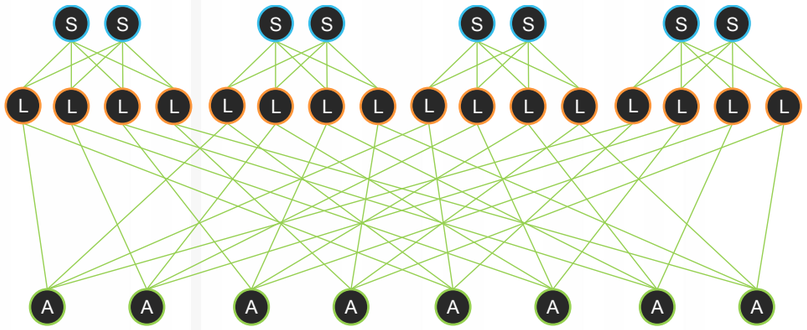
3-Stage Clos
Spine and Leaf



5-Stage Clos
PODs



5-Stage Clos
Performance



Topologie flexible en fonction des taux d'oversubscription
1 Subnet par rack (leaf/access node)

Besoin de choisir un protocole de routage
Et de quelques outils (plug&play ?)

Pourquoi du L3 jusqu'au ToR?

- MSDC (Datacenter de très grande capacité)
- Nouvelles applications "scale-out"
  - Hadoop cluster
  - Nouvelles applications "cloud" distribuées
- Nouvelle repartition de trafic (Est/Ouest)
- Convergence sur Ethernet et vers IP maintenant
  - Stockage base sur IP ("block" sur IP, système de fichiers distribués, stockage objet)
  - Overlay networking
- L3 scale très bien (partage de charge, agrégation, hiérarchie)
- Limitation des effets de bord L2 (ex: tempête broadcast, "blast radius")
- Basé sur un standard bien connu & largement inter-operable (multi vendeurs, multi chip, multi-protocoles)

Choix d'un protocole de routage

- "Simple" & well known protocol
- Robust & scalable (to support large scale "clusters")
- Easy to understand & debug
- Multi-vendor
- Flexible & Extensible

BGP

⇒ Been around "forever"

⇒ Powers Internet

⇒ Rib in // Rib out

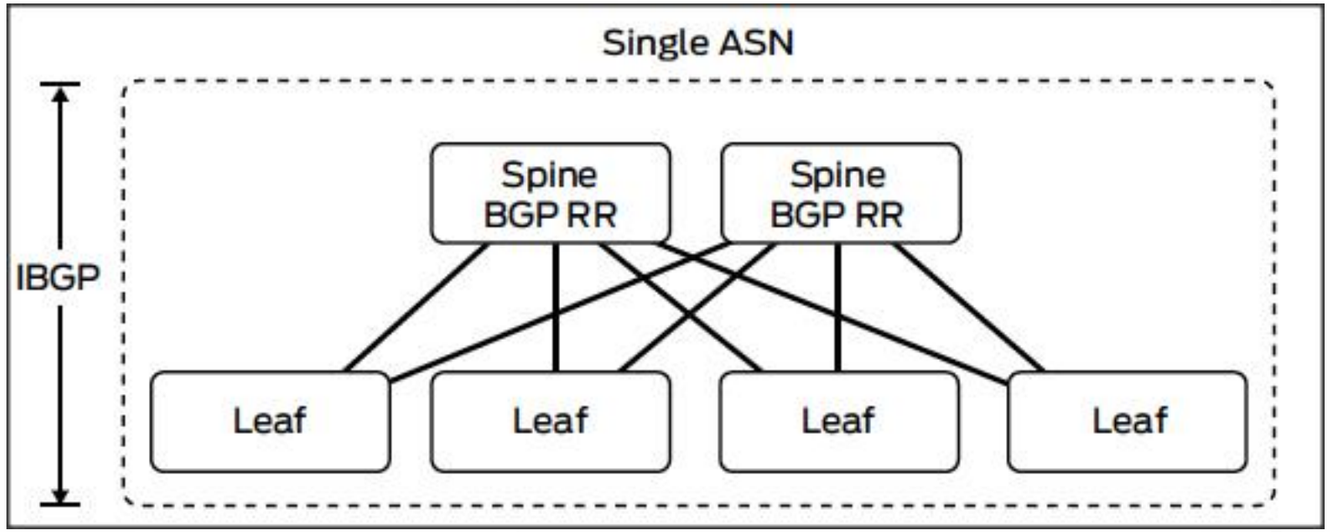
⇒ Internet again…

⇒ more on this later

Requirement	OSPF	IS-IS	BGP
Advertise prefixes	Yes	Yes	Yes
Scale	Limited	Limited	Yes
Traffic Engineering	Limited	Limited	Yes
Traffic Tagging	Limited	Limited	Yes
Multi-Vendor Stability	Yes	Yes	Even more so

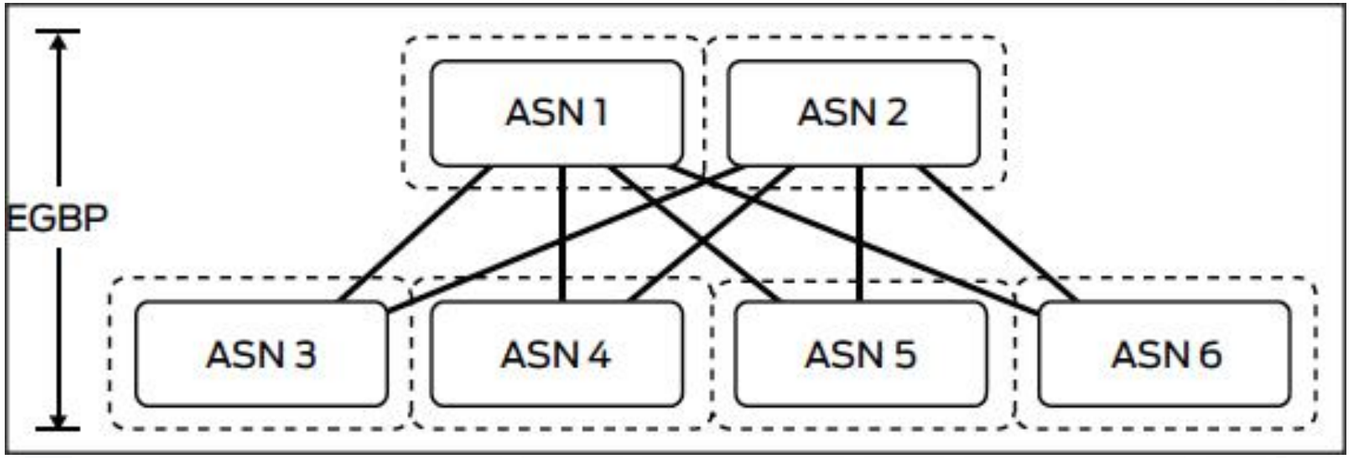
Options : E/I-BGP

i-BGP



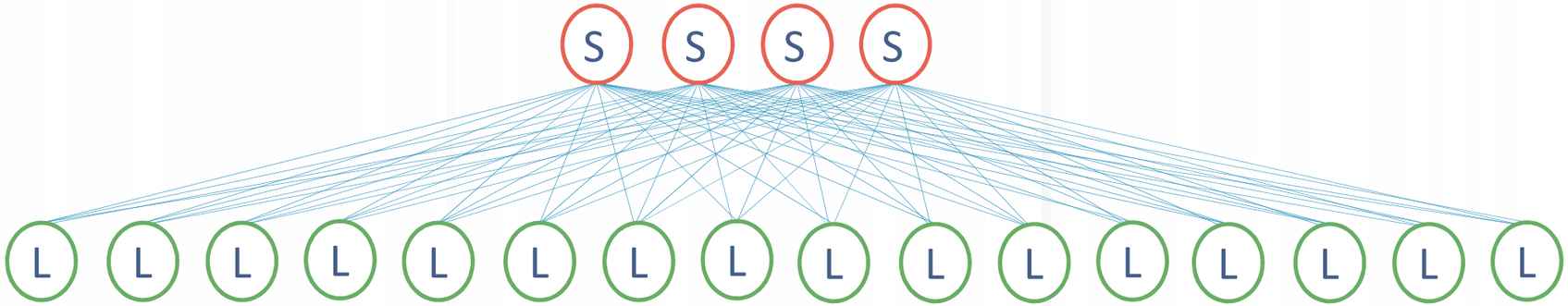
- Tendance naturelle (1 AS)
- Besoin de RR pour scale/simplification
- BGP add-path pour RIB
- ECMP sur FIB

e-BGP



- 1 AS par Switch (privee)
- Multipath “multi-AS” pour RIB
- ECMP sur FIB
- Gestion plus “naturelle” des policies
- (Traffic engineering)

Que faut-il configurer?



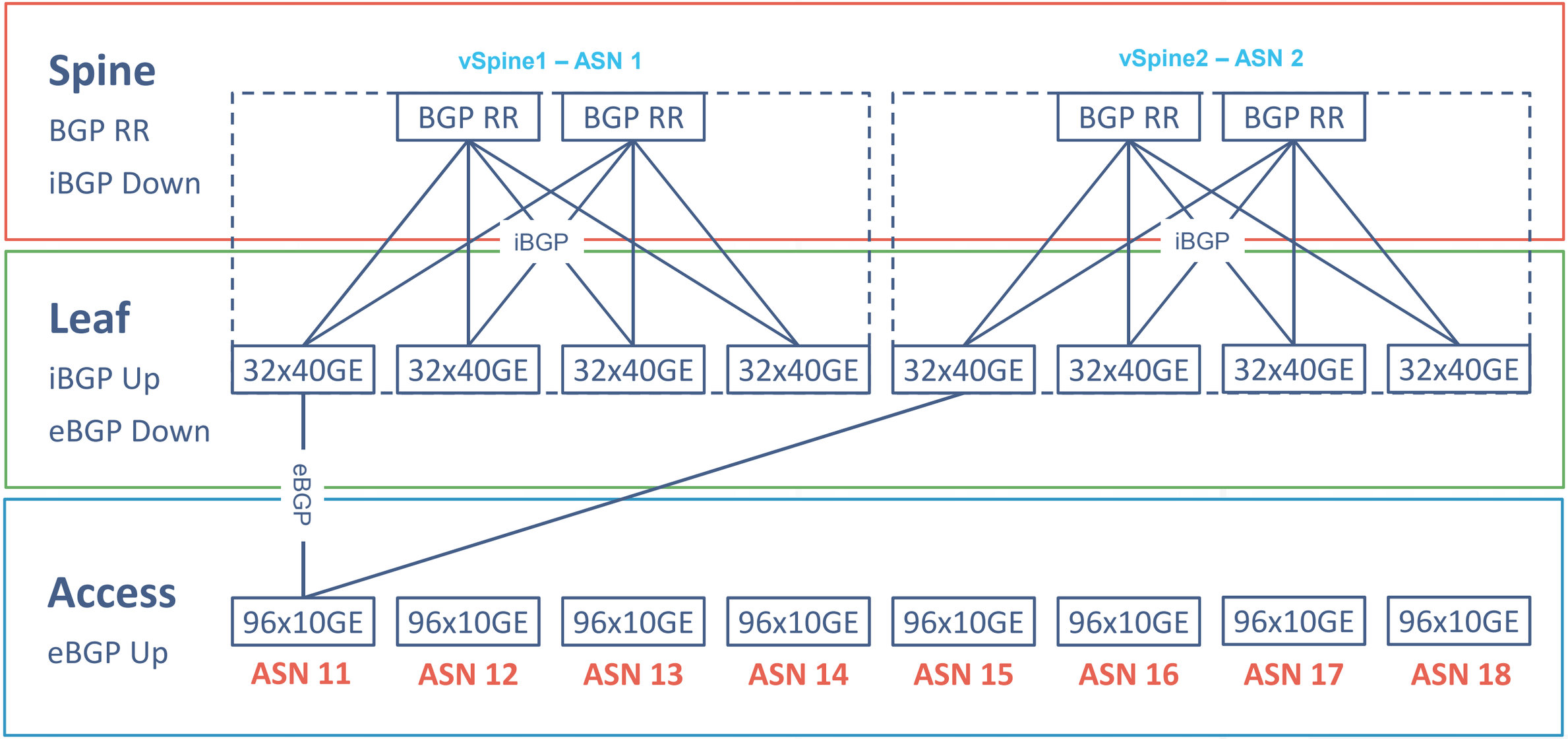
4 spines
+
16 leaves

64 links (ie P2P Networks)
+ 128 Interfaces IP
(Just infrastructure)

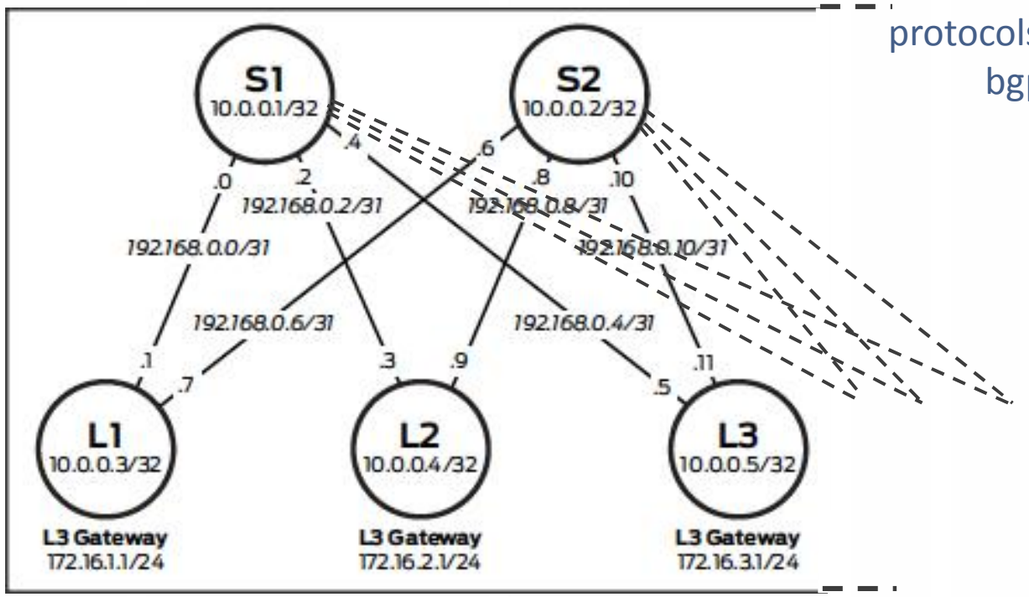
- BGP ASN assignments
- IP address scheme
  - P2P network assignments
  - P2P address assignments
  - Server-facing network assignments
  - Loopbacks
- BGP import/export policies
- Couple of knobs (ECMP)

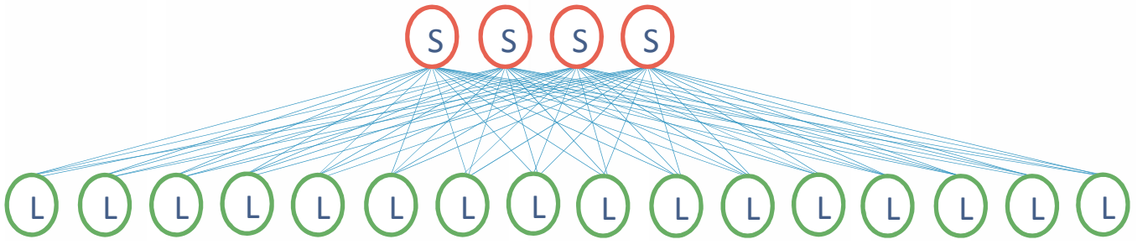
...for each and every ToR...

Exemple détaillé d'utilisation BGP jusqu'au ToR



Exemple de configuration…





4 spines
+
16 leaves

64 links (ie P2P Networks)
+ 128 Interfaces IP
(Just infrastructure)

```
protocols {
    bgp {
        log-updown;
        import bgp-clos-in;
        export bgp-clos-out;
        graceful-restart;
        group CLOS {
            type external;
            mtu-discovery;
            bfd-liveness-detection {
            minimum-interval 350;
            multiplier 3;
            session-mode single-hop;
            }
            multipath multiple-as;
            neighbor 192.168.0.1 {
                peer-as 200;
            }
            neighbor 192.168.0.3 {
                peer-as 201;
            }
            neighbor 192.168.0.5 {
                peer-as 202;
            } …
```

```
neighbor 192.168.3.7 {
    peer-as 303;
}
neighbor 192.168.3.9 {
    peer-as 304;
}
neighbor 192.168.3.11 {
    peer-as 305;
}
neighbor 192.168.3.13 {
    peer-as 306;
}
neighbor 192.168.3.15 {
    peer-as 307;
}
neighbor 192.168.3.17 {
    peer-as 308;
}
neighbor 192.168.3.19 {
    peer-as 309;
}
neighbor 192.168.3.21 {
    peer-as 310;
} …
```

```
neighbor 192.168.10.1 {
    peer-as 1001;
}
neighbor 192.168.10.2 {
    peer-as 1002;
}
neighbor 192.168.10.3 {
    peer-as 1003;
}
neighbor 192.168.10.4 {
    peer-as 1004;
}
neighbor 192.168.10.5 {
    peer-as 1005;
}
neighbor 192.168.10.6 {
    peer-as 1006;
}
neighbor 192.168.10.7 {
    peer-as 1007;
}
neighbor 192.168.10.8 {
    peer-as 1008;
} …
```

Automatisation - Outils

- Besoin de générer automatiquement la configuration
  - Liste de variables + Topologie (ASN, Subnet – dans fichier YAML/Script)
  - Python avec Template (jinja2)
- Chargement “initial” de la configuration
  - Zero Touch Provisioning (ZTP)
- Ou utilisation des interfaces de configuration du switch
  - Netconf / XML API
  - Python wrapper : PythonEZ mini-framework
  - Ansible
- Outil pour valider la configuration et l'état opérationel de l’infrastructure
  - PythonEZ // JSNAP => Utilisation aisée de la structure XML
  - BMP/SNMP
- Outils opensource - Github

Conclusion

- 1ère utilisation décrite dans la présentation : routage IP dans l'infrastructure jusqu'au ToR
  - Simple, scalable, robuste, inter-operable
  - Et extensible
- Il y a bien d'autres avantages à utiliser BGP:
  - Routage dans le serveur, multihoming/anycast (Quagga)
  - Echange d'états L2 (MAC Address): EVPN (over VXLAN ou MPLS)
    - Création d'Overlay dans la Fabric (L2 sur Fabric L3)
  - Echange d'états entre un réseau SDN/Overlay & l'Underlay (infra)
    - Le contrôleur peer en BGP vers le réseau physique
    - Les "routes (L2/L3)" sont échangées ET contrôlées
- En utilisant des standards
  - VPN-IP routes (rfc4364 aka 2547 VPNs)
  - MAC routes (draft-ietf-l2vpn-evpn)

Questions?

asibout@juniper.net

MULTI-STAGE CLOS ROLES

Spine

- Backplane of multi-stage CLOS
- Always 1:1 Over-Subscription
- Provide BGP Route Reflection
- Peers via iBGP to Leaf nodes

Leaf

- NNI of multi-stage CLOS
- Variable Over-Subscription
- Peers via iBGP to Spine nodes
- Peers via eBGP to Access nodes

vSpine

- Combination of Spine and Leaf
- Acts as a logical switch
- Virtual peering point for Access
- Over-Subscription dependent on the Spine and Leaf roles
- Single BGP Autonomous System Number
- Peers via eBGP to access switches

Access

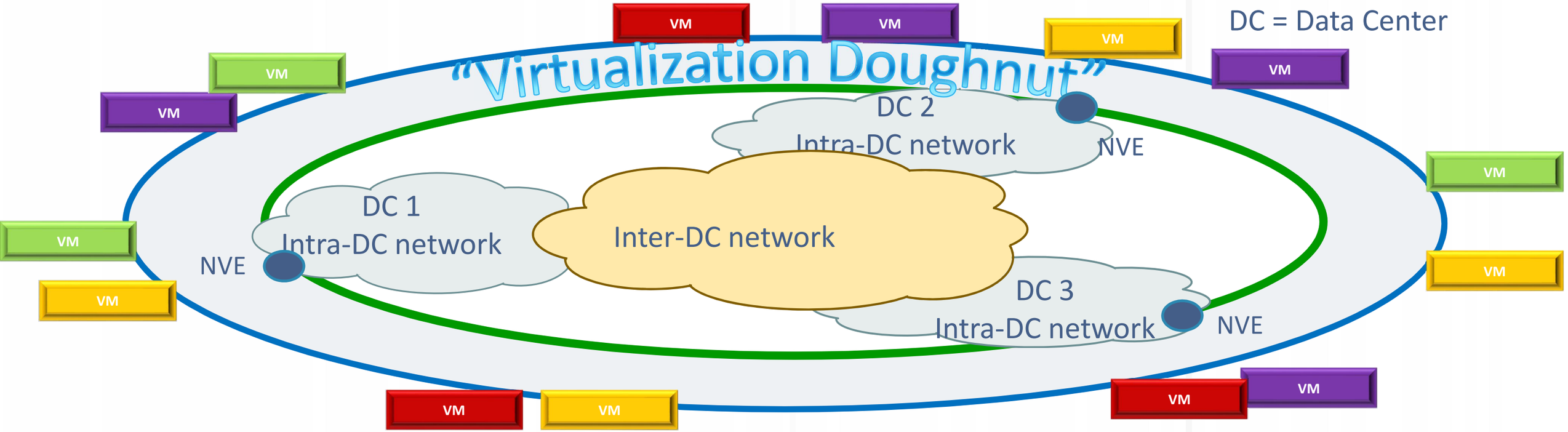
- Provide access to end-points such as compute and storage
- Typically 3:1 Over-Subscription in ENT and SP environments, and 1:1 for HPC
- Peers via eBGP to vSpine nodes
- Provides L3 gateway services to end-points
- Provides Link Aggregation to end-points

SDN/Overlay - NVo3 & Doughnuts

State management for Cloud computing service

- Cloud Computing service requires managing (very) large amount of state
  - Due to the need to support (very) large number of customers (aka “tenants”), with multiple Virtual Machines (VMs) per customer
    - Driven by the need to take advantage of the economy of scale
- Cloud Computing service requires managing (fairly) dynamic state
  - Due to creation, termination, and moves of VMs
    - Driven by the need to support fast on-demand delivery of required Cloud Computing service
    - Driven by the need to optimize usage of resources (e.g., servers) used by Cloud Computing service
- Scalability, **both** in terms of the total amount of state, **and** in terms of the state’s dynamic is of critical importance
- *Note: In the rest of the slides the term “state used by Cloud Computing services” refers to the state associated with VMs of individual customers/tenants of Cloud Computing service*

Scalability via “Virtualization doughnut”



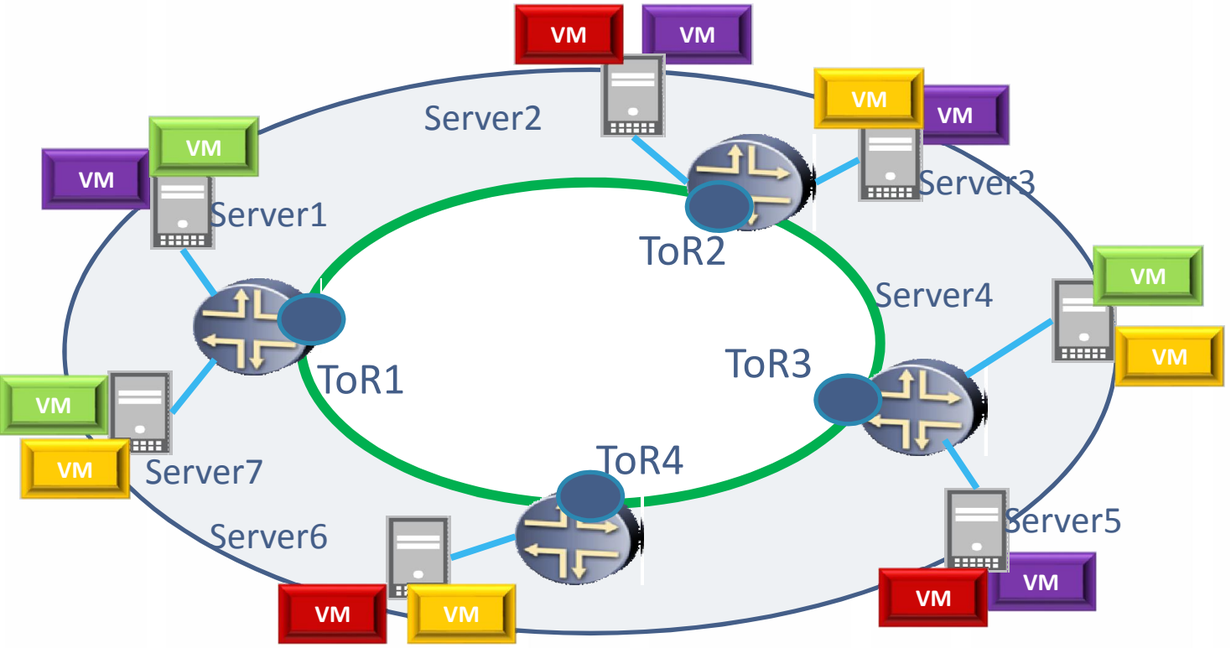
- VMs on the outside: connected to the outer virtualization edge
- Intra-DC/Inter-DC networks on the inside: connected to the inner virtualization edge – Network Virtualization Edge devices (NVEs)
- NVEs use some form of tunnel encapsulation to form an overlay over the network infrastructure that interconnects them (just like PEs in 2547 VPN)

What is in the doughnut ?

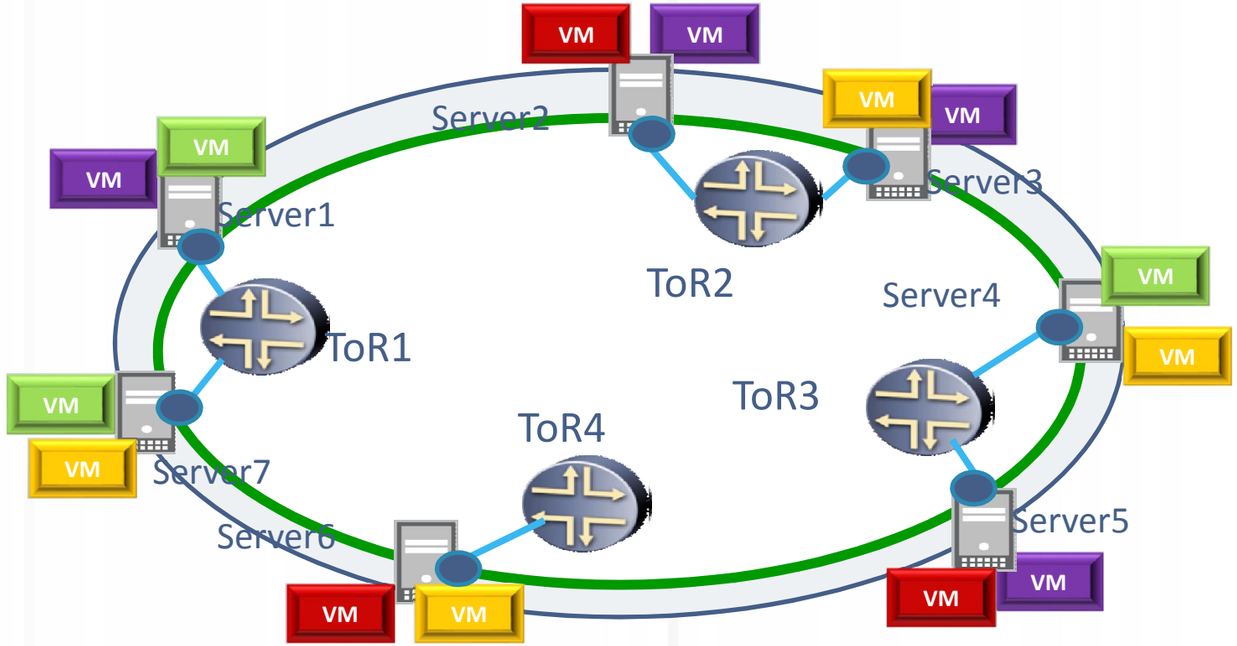
Virtualization doughnut has to include servers -> servers must be in the doughnut

Where are ToR switches ? In or out of the doughnut ? Where is NVE ?

THICK DOUGHNUT



THIN DOUGHNUT

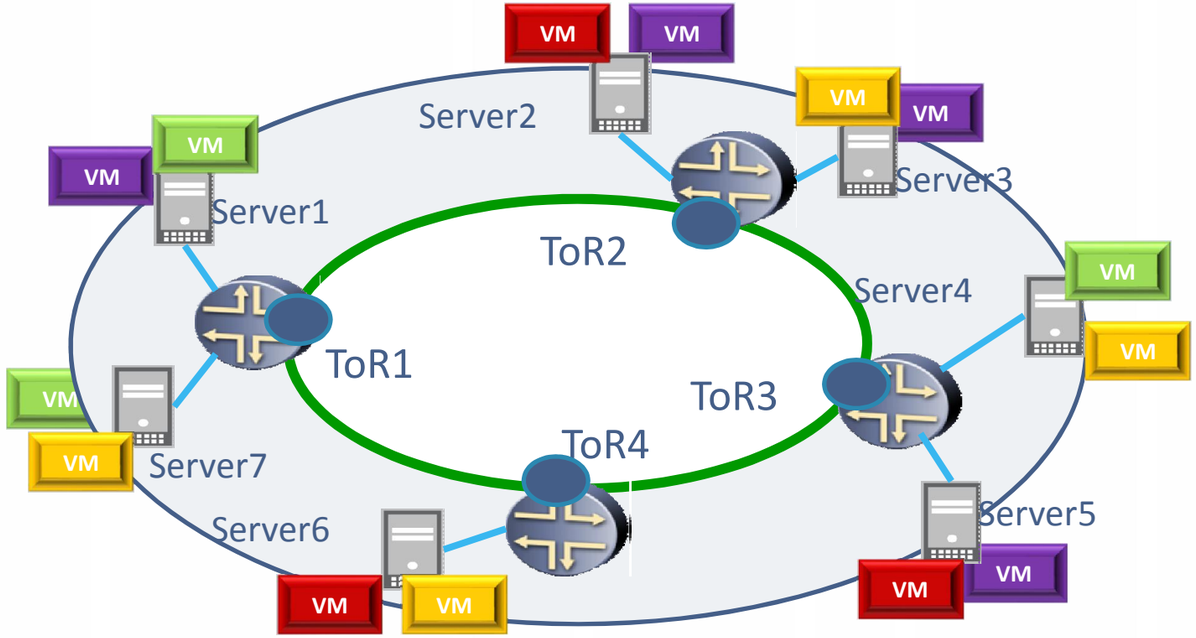


Thick doughnut places NVEs on ToRs
-> <u>ToRs are part the doughnut</u>

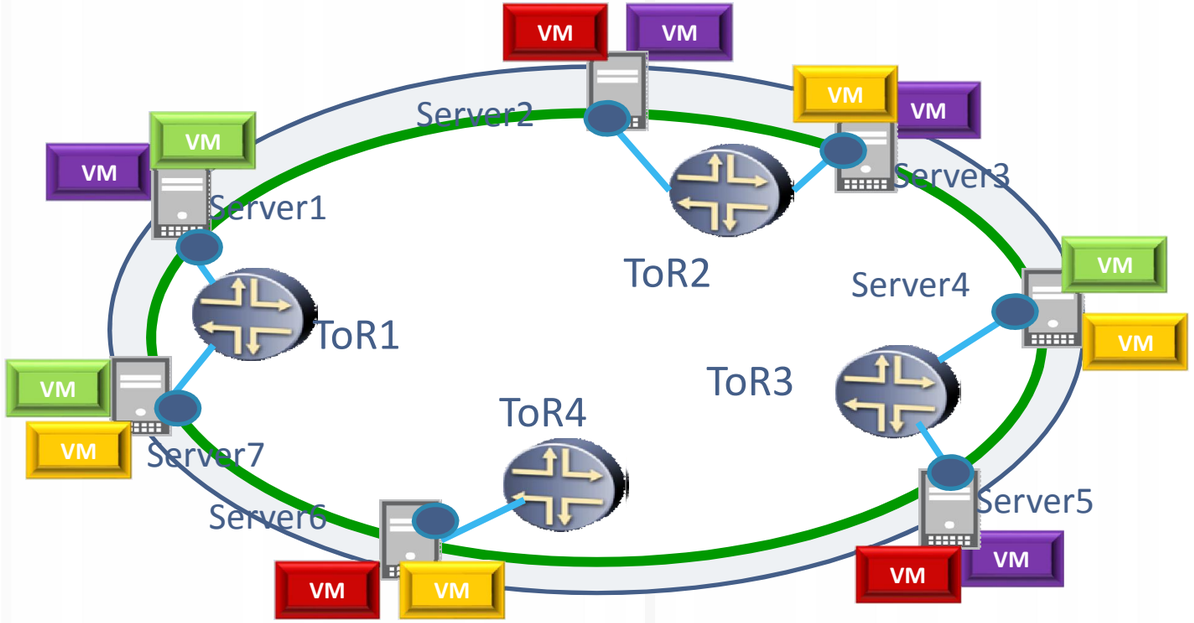
Thin doughnut places NVEs on servers
-> <u>ToRs are NOT part of the doughnut</u>

Thick vs Thin doughnut

THICK DOUGHNUT



THIN DOUGHNUT



- In thick doughnut service creation /management requires coordination between servers and ToRs
- NVE on ToR has to maintain state for VMs of a given customer/tenant, as long as the customer/tenant has at least one of these VMs hosted <u>on any of the servers connected to that ToR</u>

- No need for orchestration
- NVE on server has to maintain state for VMs of a given customer/tenant, as long as the customer/tenant has at least one of these VMs hosted <u>on that server</u>

Thin doughnut allows to improve scalability by reducing the amount of state that an individual NVE has to maintain (relative to thick doughnut)

Distributed State management with BGP

- Scalability via state management hierarchy:
  - state management endpoint need not have control plane peering with every other state management endpoint
  - State management hierarchy MUST NOT force hierarchy on the data plane
    - Important for efficient use of resources
- BGP supports state management hierarchy by using BGP Route Reflectors
  - State management endpoint need not have BGP peering with every other state management endpoint – only with few BGP Route Reflectors
    - State management endpoints form the bottom level of the hierarchy, BGP Router Reflectors form the top level of the hierarchy
  - Use of BGP “third-party” Next Hop allows to impose state management hierarchy **without** forcing hierarchy on the data plane
  - Additional levels of state management hierarchy could be introduced by forming hierarchy of BGP Route Reflectors

Scalability via “divide and conquer”

- Scalability via “divide and conquer”:
  - Distribution scope of state information associated with creation, termination, and moves of a given VM should be constrained to only the NVEs that are connected to VMs of a single Cloud Computing service customer
    - Customer who “owns” this VM
- BGP can constrain distribution scope of state information by using Route Targets and Route Target Constrains (rfc4684)
  - NVE uses Route Target Constrains to inform its Route Reflector(s) of the customers for which it needs to receive state information
  - Route Reflector uses the Route Target Constrains information it receives from an NVE to send to that NVE only the state information of the customers that have at least one of their VMs connected to that NVE

Scalability via “divide and conquer” and State Management hierarchy

“Divide and conquer” could be used in combination with state management hierarchy :

- Use BGP Route Reflectors for state management hierarchy
- Partition BGP Route Reflectors within the Cloud Computing service provider among Cloud Computing service customers served by the Provider

- As a result:
  - No single component within the system (neither NVEs, nor Route Reflectors) is required to maintain all state for all the VMs
  - Capacity of the system is **not** bounded by the capacity of its individual components

BGP: wide variety of state information

- BGP can distribute a wide variety of state information:
  - State associated with IP routes (rfc4271)
  - State associated with VPN-IP routes (rfc4364 aka 2547 VPNs)
  - State associated with MAC routes (draft-ietf-l2vpn-evpn)
  - State associated with RT Constrain Routes (rfc4684)
  - State associated with Flow Spec (rfc5575)
  - State associated with multicast routing information (rfc6514)
- All of the above are either IETF standards, or on the IETF standards track
- All of the above may be applicability in the context of state management for Cloud Computing service
- Single tool for distributing multiple types of state information