Università degli Studi Roma Tre
Dipartimento di Ingegneria
Computer Networks Research Group

data centers and VXLAN

Version	3.1
Author(s)	L. Ariemma, G. Di Battista, M. Patrignani, M. Scazzariello, T. Caiazzi
E-mail	contact@kathara.org
Web	http://www.kathara.org/
Description	Data Centers' Routing: BGP and VXLAN

copyright notice

- all the pages/slides in this presentation, including but not limited to, images, photos, animations, videos, sounds, music, and text (hereby referred to as “material”) are protected by copyright
- this material, with the exception of some multimedia elements licensed by other organizations, is property of the authors and/or organizations appearing in the first slide
- this material, or its parts, can be reproduced and used for didactical purposes within universities and schools, provided that this happens for non-profit purposes
- any other use is prohibited, unless explicitly authorized by the authors on the basis of an explicit agreement
- this copyright notice must always be redistributed together with the material, or its portions

how to handle multiple tenants?

again, why BGP?

overview – multiple tenants

- requirements
  - servers' architecture requirements
  - orchestration requirements
  - tenant requirements
- tunneling protocols
- VXLAN
- EVPN-BGP

servers' architecture requirements

- having services directly on bare metal is not used
  - too many physical servers needed
  - no way to scale if more resources are requested
  - no isolation between different services on the same server
- support a virtual layer of containers or VMs
  - high-availability guaranteed via orchestration
  - useful for resource-slicing
  - complete isolation between different containers or VMs on the same server
    - possibility to assign containers or VMs to different tenants

orchestration requirements

- an orchestrator is a software that manages the lifecycle of containers/VMs
  - creates, moves, and destroys containers/VMs
- needed for
  - optimal resource allocation, handling failures, management
- when moving a container/VM
  - possibility to keep network configurations (MAC/IP)
  - minimal downtime

tenant requirements

- each tenant wants to independently manage its own private IP address space
  - containers/VMs traffic must be segregated between tenants and between the data center traffic

consequence of requirements

- server, orchestration, and tenant requirements have a consequence
  - usage of tunnels

tunneling protocol requirements

- minimal configuration
- encapsulate the traffic of each tenant
  - an identifier of the tenant is needed
- encapsulation in Layer-4
  - to fully exploit IP Multi-Path
  - to traverse routers
    - data center fabric is Layer-3
  - to traverse Internet
    - different data centers must interconnect via Internet transparently (to create the so called *regions*)

possible choices for tunneling

- VLAN
  - can be used only in L2, the data center is L3
- MPLS
  - each router must be configured for each new tenant
  - traversing Internet requires ISP to configure intermediate routers
- VXLAN
  - created ad-hoc ☺

VXLAN

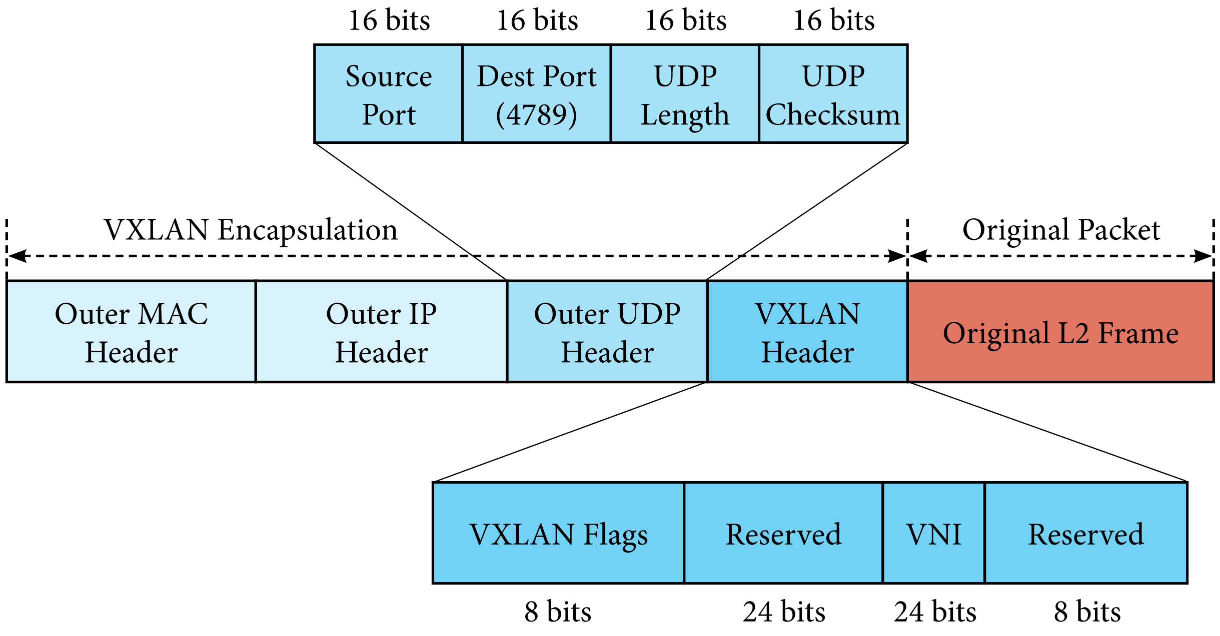
- Virtual eXtensible Local Area Network (RFC-7348)
- designed to address the need for overlay networks within virtualized data centers accommodating multiple tenants
- encapsulates Layer-2 frames into UDP packets

VXLAN terminology

- VNI: VXLAN Network Identifier
  - identifier of a specific VXLAN tunnel
  - similar to the VLAN ID
  - 24 bit address space, more than 16M possible VNIs
- VTEP: VXLAN Tunnel End Point
  - device (physical or virtual) that encapsulates and decapsulates VXLAN packets

VXLAN encapsulation

- overhead of 50 bytes
- random Source Port to fully exploit Multi-Path



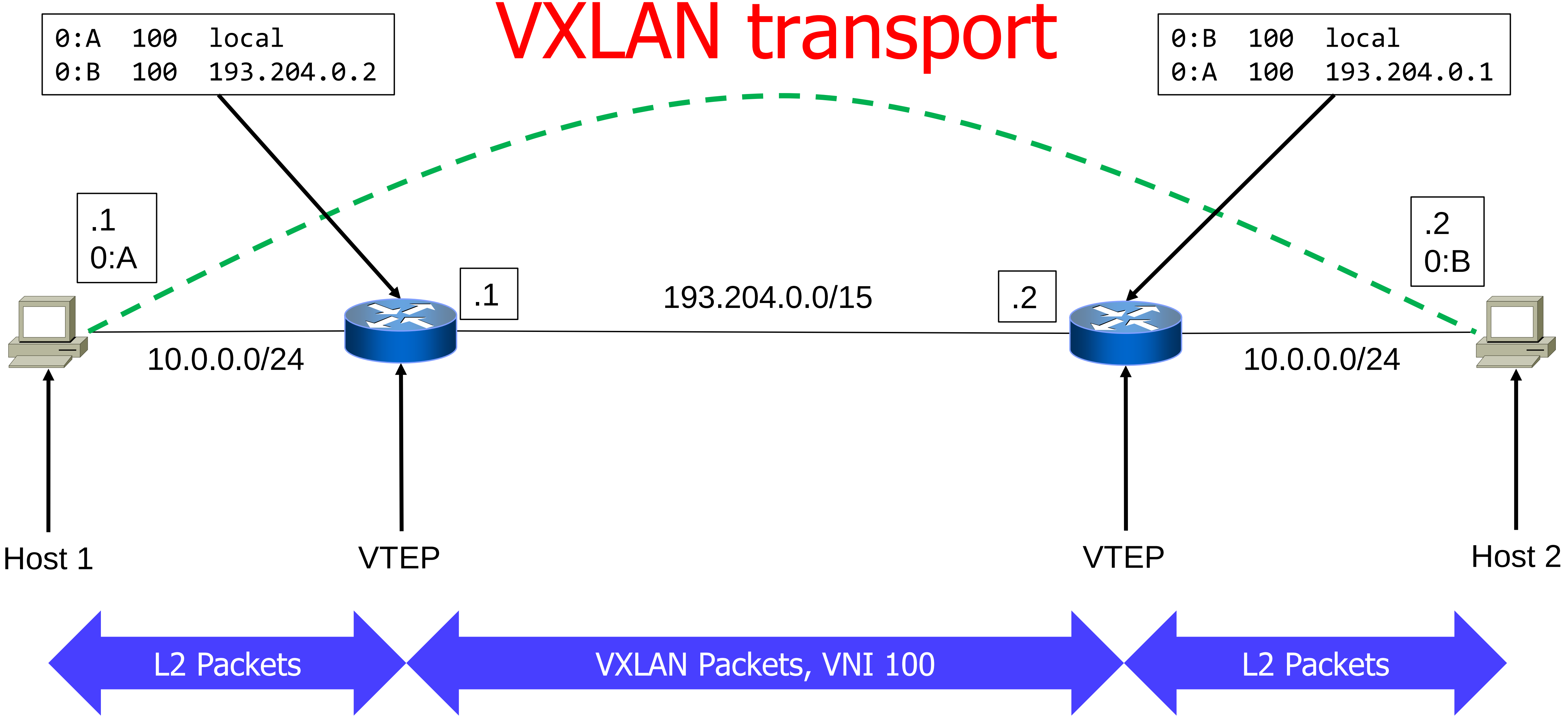
MAC-to-VTEP Table

- hosted in each VTEP
- similar to the switch forwarding table
- for each VNI the VTEP keeps a table of pairs `<`*`mac,ip`*`>`, that associates MAC Addresses and destination VTEP IPs
- each physical (or VLAN) L2 interface of a VTEP is assigned to a VNI
  - the MAC addresses learned on such interfaces are *local,* and the IPs of their pairs is replaced by the word *local*

MAC-to-VTEP Table

- when a VTEP receives a frame destinated to a MAC Address m from a local interface belonging to a certain VNI, it checks for the existence (in the VNI) of a pair $<m, i>$ containing m
  - if the pair exists, the VTEP encapsulates the frame, and sends the resulting packet to the destination VTEP IP i
  - if not, the frame is sent to all the other VTEPs (encapsulated) and to all the local ports of that VNI

VXLAN transport



how to handle broadcast traffic

- two types of broadcast must be handled
  - traffic directed to the MAC broadcast address
    - e.g., ARP traffic
  - traffic directed to a MAC address that has not been learned
- IP multicast groups are used by default
  - each VNI is assigned to an IP multicast group of the underlay network and each VTEP subscribes itself to each group of its VNIs
  - the multicast group of a VNI is used only to send the broadcast traffic

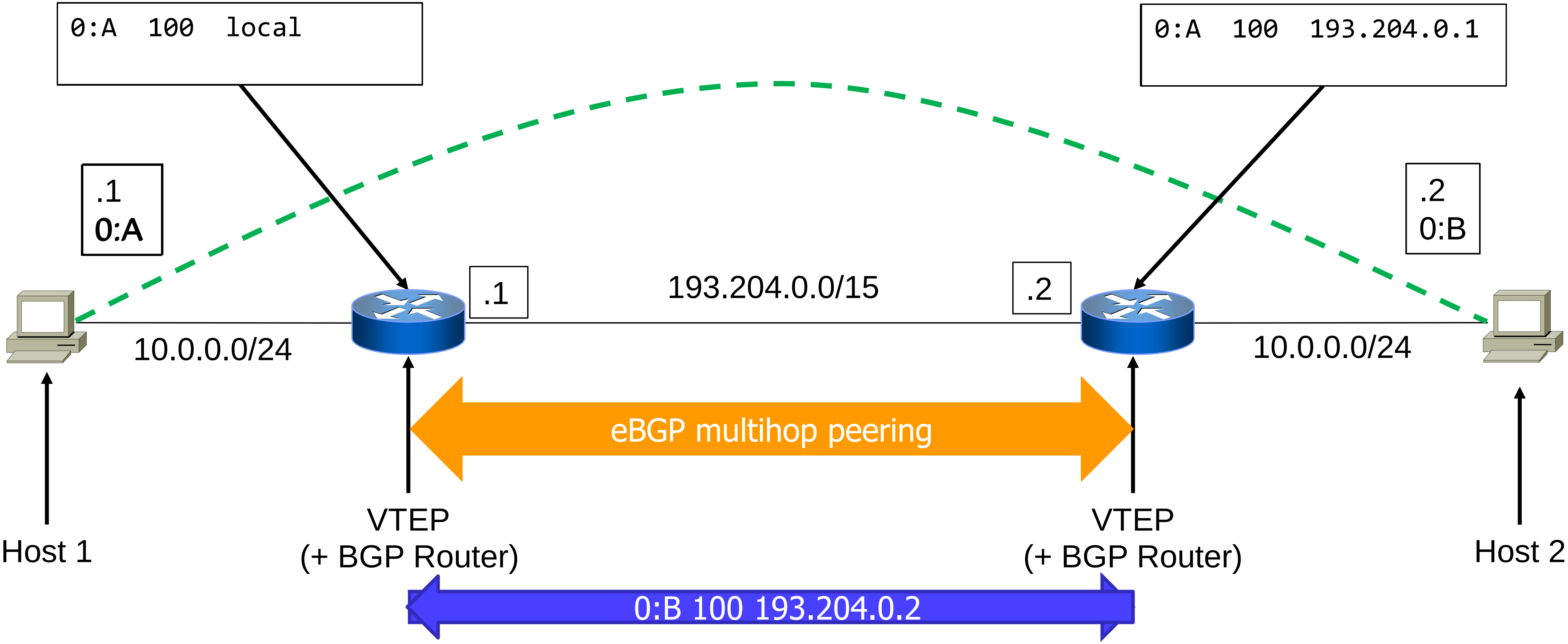
disadvantages of multicast and alternatives

- multicast must be enabled in the underlay network
  - it may require to deploy several protocols
    - e.g., IGMP, IGMP Snooping, PIM, ….
  - complex configuration
- if multicast is not enabled, broadcast frames are duplicated and sent unicast to all the VTEPs of the VNI
- proxy ARP techniques can be used to mitigate broadcast traffic

EVPN-BGP

- Ethernet VPN (RFC-7432 and RFC-8365)
- uses MP-BGP with specific AFI/SAFI
  - Address Family Identifier/Subsequent AFI
  - AFI=25 (L2VPN) – SAFI=70 (EVPN)
- advertises MAC Addresses of VNIs using BGP updates
- a VTEP automatically learns local MAC Addresses and advertises them to all other VTEPs (of the same VNI)
- VTEP proxies ARP requests to limit broadcast traffic

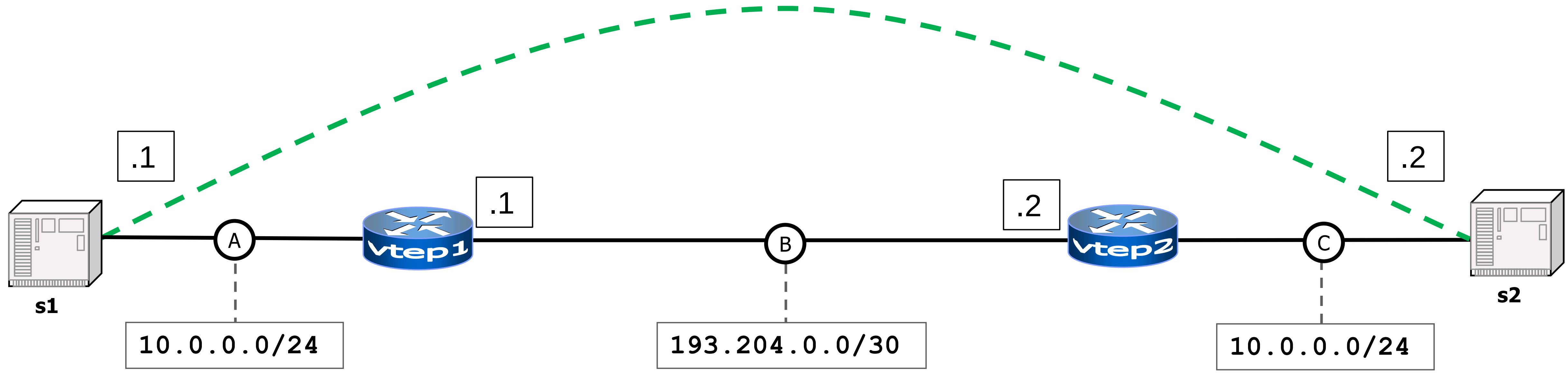
VXLAN and EVPN-BGP transport



VXLAN and EVPN-BGP Lab

time to use Kathará

topology



lab base config – topology, s1, and s2

lab.conf

```
s1[0]=A

vtep1[0]=A
vtep1[1]=B

vtep2[0]=C
vtep2[1]=B

s2[0]=C
```

s1.startup

```
ifconfig eth0 10.0.0.1/24 up
ip link set dev eth0 mtu 1450
/etc/init.d/apache2 start
```

s2.startup

```
ifconfig eth0 10.0.0.2/24 up
ip link set dev eth0 mtu 1450
/etc/init.d/apache2 start
```

MTU

- need to set manual MTU of each device's interface associated to a VNI
- when a frame is encapsulated by a VTEP, if its size is greater than 1450 (LAN MTU – VXLAN overhead)
  - the frame is dropped
  - an ICMP "packet too big" message is sent back

configuring a bridge/router with VTEPs

- we need to configure a Kathara device that is able to:
  - act as a bridge on L2 ports
    - able to perform L2 learning
  - act as a router on the ports on the underlay networks
    - able to establish BGP peerings
    - able to encapsulate/decapsulate VXLAN traffic

VTEP configuration

- create bridge facilities (aka companion bridges)
  - one virtual bridge with ports assigned to VLANs corresponding to VNIs
- attach the collision domains associated to a VNI to the companion bridge of that VNI
- configure the base BGP peerings
  - enable the AFI/SAFI of EVPN
- configure VXLAN

vtep e-BGP configuration

vtep1/etc/frr/bgpd.conf

```
router bgp 1

neighbor 193.204.0.2 remote-as 2

address-family l2vpn evpn
    neighbor 193.204.0.2 activate
    advertise-all-vni
exit-address-family
```

vtep2/etc/frr/bgpd.conf

```
router bgp 2

neighbor 193.204.0.1 remote-as 1

address-family l2vpn evpn
    neighbor 193.204.0.1 activate
    advertise-all-vni
exit-address-family
```

activate the address-family on the peering with specific neighbor

announce on the peerings all the VNIs configured in the VTEP

companion bridge

- the bridge connected to VTEP interfaces
- used by the VTEP to perform source address learning of local interfaces
- when in combination with EVPN-BGP
  - its forwarding table is also populated via updates received from BGP
  - the FRR control plane watch to updates of the bridge forwarding table to send updates via BGP

vtep1 bridge configuration

vtep1.startup

```
ifconfig eth1 193.204.0.1/30 up

# Setting up VXLAN interfaces
ip link add vtep100 type vxlan id 100 dev eth1 dstport 4789 local 193.204.0.1 nolearning
ip link set up dev vtep100

# Creating the companion bridge
ip link add br100 type bridge
# Attach interfaces to the bridge
ip link set dev vtep100 master br100
ip link set dev eth0 master b
# Enable bridge vlans
ip link set dev br100 type bridge vl
bridge vlan add vid 10 untagged
bridge vlan add vid 100 dev eth0 pvid untagged
ip link set up dev br100

# Enabling FRR
/etc/init.d/frr start
```

VNI

create the companion bridge and name it

port

attach the VXLAN interface to the bridge

VXLAN src IP

name of the interfac

enable the VXLAN interface

inter
VXL
the u

attach eth0 (the server s1 collision domain) to the bridge

disable the mcast learning

create the interface

enable VLANs on the bridge

configure VLANs on bridge ports

enable the bridge

vtep2 bridge configuration

vtep2.startup

```
ifconfig eth1 193.204.0.2/30 up

# Setting up VXLAN interfaces
ip link add vtep100 type vxlan id 100 dev eth1 dstport 4789 local 193.204.0.2 nolearning
ip link set up dev vtep100

# Creating the companion bridge
ip link add br100 type bridge
# Attach interfaces to the bridge
ip link set dev vtep100 master br100
ip link set dev eth0 master br100
# Enable bridge vlans
ip link set dev br100 type bridge vlan_filtering 1
bridge vlan add vid 100 dev vtep100 pvid untagged
bridge vlan add vid 100 dev eth0 pvid untagged
ip link set up dev br100

# Enabling FRR
/etc/init.d/frr start
```

the EVPN-BGP control-plane

```
vtep1# show evpn mac vni all

VNI 100 #MACs (local and remote) 2

Flags: N=sync-neighs, I=local-inactive, P=peer-active, X=peer-proxy
MAC                  Type    Flags Intf/Remote ES/VTEP            VLAN  Seq #'s
a2:6d:75:c7:06:6f remote       193.204.0.2                              0/0
c2:93:31:36:31:b6 local        eth0                                     0/0
```

a BGP update

```
> Frame 3: 170 bytes on wire (1360 bits), 170 bytes captured (1360 bits)
> Ethernet II, Src: ee:8d:53:3a:12:b5 (ee:8d:53:3a:12:b5), Dst: ea:4f:03:8c:4b:95 (ea:4f:03:8c:4b:95)
> Internet Protocol Version 4, Src: 193.204.0.2, Dst: 193.204.0.1
> Transmission Control Protocol, Src Port: 41530, Dst Port: 179, Seq: 1, Ack: 1, Len: 104
v Border Gateway Protocol - UPDATE Message
    Marker: ffffffffffffffffffffffffffffffff
    Length: 104
    Type: UPDATE Message (2)
    Withdrawn Routes Length: 0
    Total Path Attribute Length: 81
  v Path attributes
    v Path Attribute - MP_REACH_NLRI
      > Flags: 0x90, Optional, Extended-Length, Non-transitive, Complete
        Type Code: MP_REACH_NLRI (14)
        Length: 44
        Address family identifier (AFI): Layer-2 VPN (25)
        Subsequent address family identifier (SAFI): EVPN (70)
      > Next hop: 193.204.0.2
        Number of Subnetwork points of attachment (SNPA): 0
      v Network Layer Reachability Information (NLRI)
        v EVPN NLRI: MAC Advertisement Route
            Route Type: MAC Advertisement Route (2)
            Length: 33
            Route Distinguisher: 0001c1cc00020002 (193.204.0.2:2)
          > ESI: 00:00:00:00:00:00:00:00:00:00
            Ethernet Tag ID: 0
            MAC Address Length: 48
            MAC Address: a2:6d:75:c7:06:6f (a2:6d:75:c7:06:6f)
            IP Address Length: 0
          > IP Address: NOT INCLUDED
            VNI: 100
    > Path Attribute - ORIGIN: IGP
    > Path Attribute - AS_PATH: 2
    > Path Attribute - EXTENDED_COMMUNITIES
```

from vtep2 to vtep1

announcement

AFI/SAFI of l2vpn/EVPN

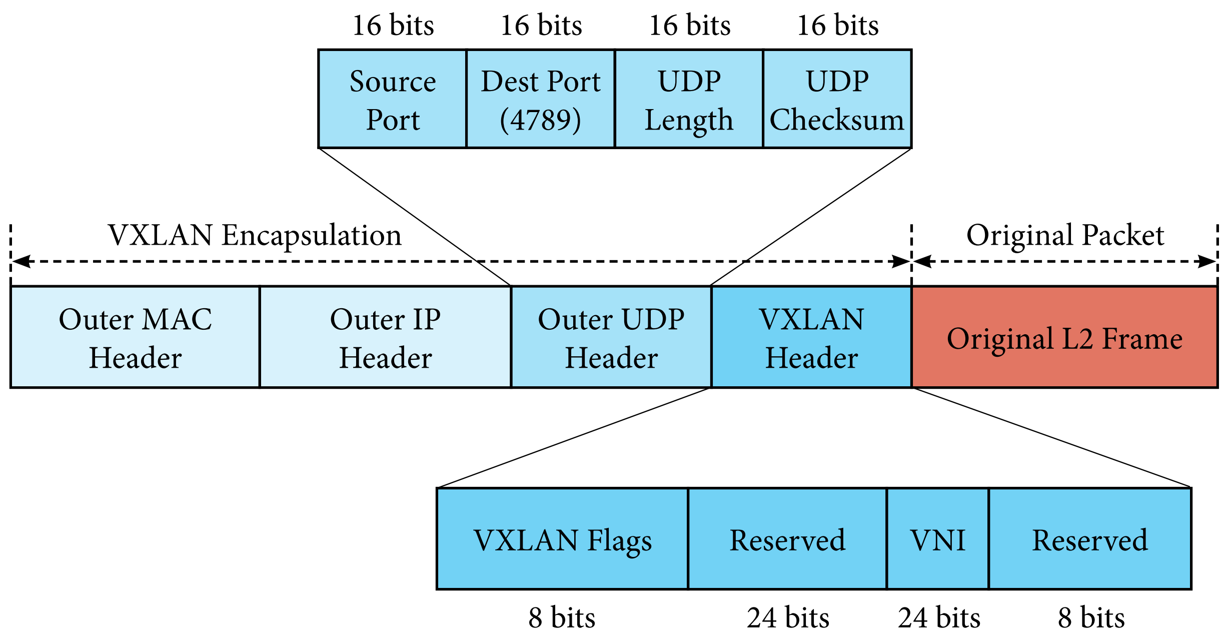
VTEP destination

MAC address of s1

VNI of s1

VXLAN encapsulation

- overhead of 50 bytes
- random Source Port to fully exploit Multi-Path



a PING packet encapsulated in VXLAN

```
> Frame 11: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits)
> Ethernet II, Src: ea:4f:03:8c:4b:95 (ea:4f:03:8c:4b:95), Dst: ee:8d:53:3a:12:b5 (ee:8d:53:3a:12:b5)
> Internet Protocol Version 4, Src: 193.204.0.1, Dst: 193.204.0.2
> User Datagram Protocol, Src Port: 38648, Dst Port: 4789
> Virtual eXtensible Local Area Network
> Ethernet II, Src: c2:93:31:36:31:b6 (c2:93:31:36:31:b6), Dst: a2:6d:75:c7:06:6f (a2:6d:75:c7:06:6f)
> Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.2
v Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x0945 [correct]
    [Checksum Status: Good]
    Identifier (BE): 11 (0x000b)
    Identifier (LE): 2816 (0x0b00)
    Sequence Number (BE): 2 (0x0002)
    Sequence Number (LE): 512 (0x0200)
    [Response frame: 12]
    Timestamp from icmp data: Dec  6, 2022 15:59:10.000000000 CET
    [Timestamp from icmp data (relative): 0.401357000 seconds]
  > Data (48 bytes)
```

Outer Layer 1

Outer MAC Header

Outer IP Layer

Outer UDP Header

VXLAN Header

Original Layer 2

Original Layer 3

Original ICMP Packet

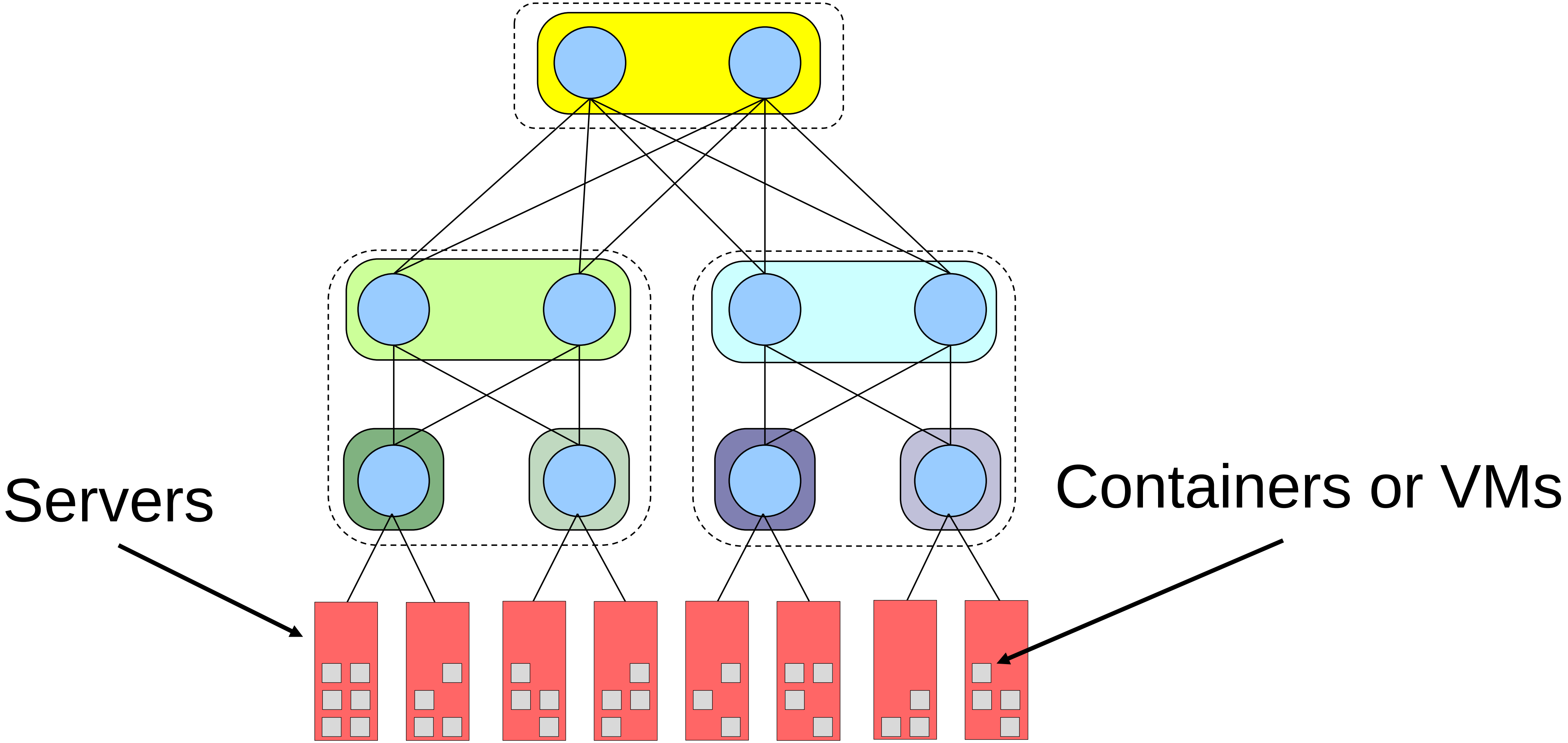
putting together

(EVPN-)BGP 😉

overview

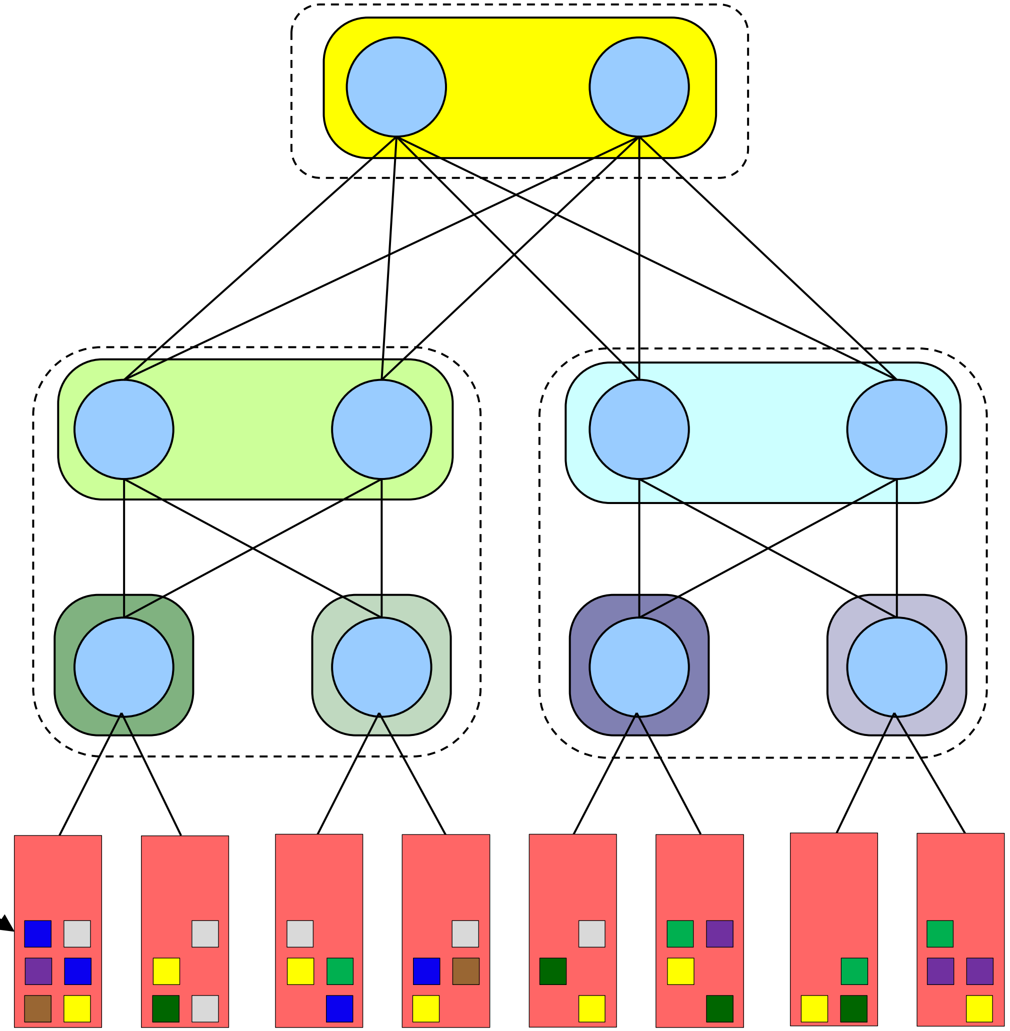
- containers or VMs of different tenants
- where is the VTEP?
- the Leaf-server links
- inside the servers
- dual attached servers
- a complete lab experience

servers in the fabric – recap



containers or VMs of different tenants

different colours represent different tenants



EVPN-BGP – where is the VTEP?

- containers must be unaware of tunneling
- different choices for positioning the VTEP
  - in each server
    - the server should have a BGP peering for enabling EVPN-BGP
    - the server CPU would be used to route packets
  - <u>in each Leaf</u>
    - a Leaf already has a BGP peering
    - a Leaf is a router, so it has dedicated routing hardware
    - usage of VLANs in the link connecting a Leaf and a server to distinguish tenants

inside the Leaves

- VLANs are used
- mapping between VNIs and VLAN IDs
  - tenants are unaware of the mapping
- Leaves decapsulate VXLAN received packets and encapsulate them into VLAN frames, according to the mapping
  - and vice-versa

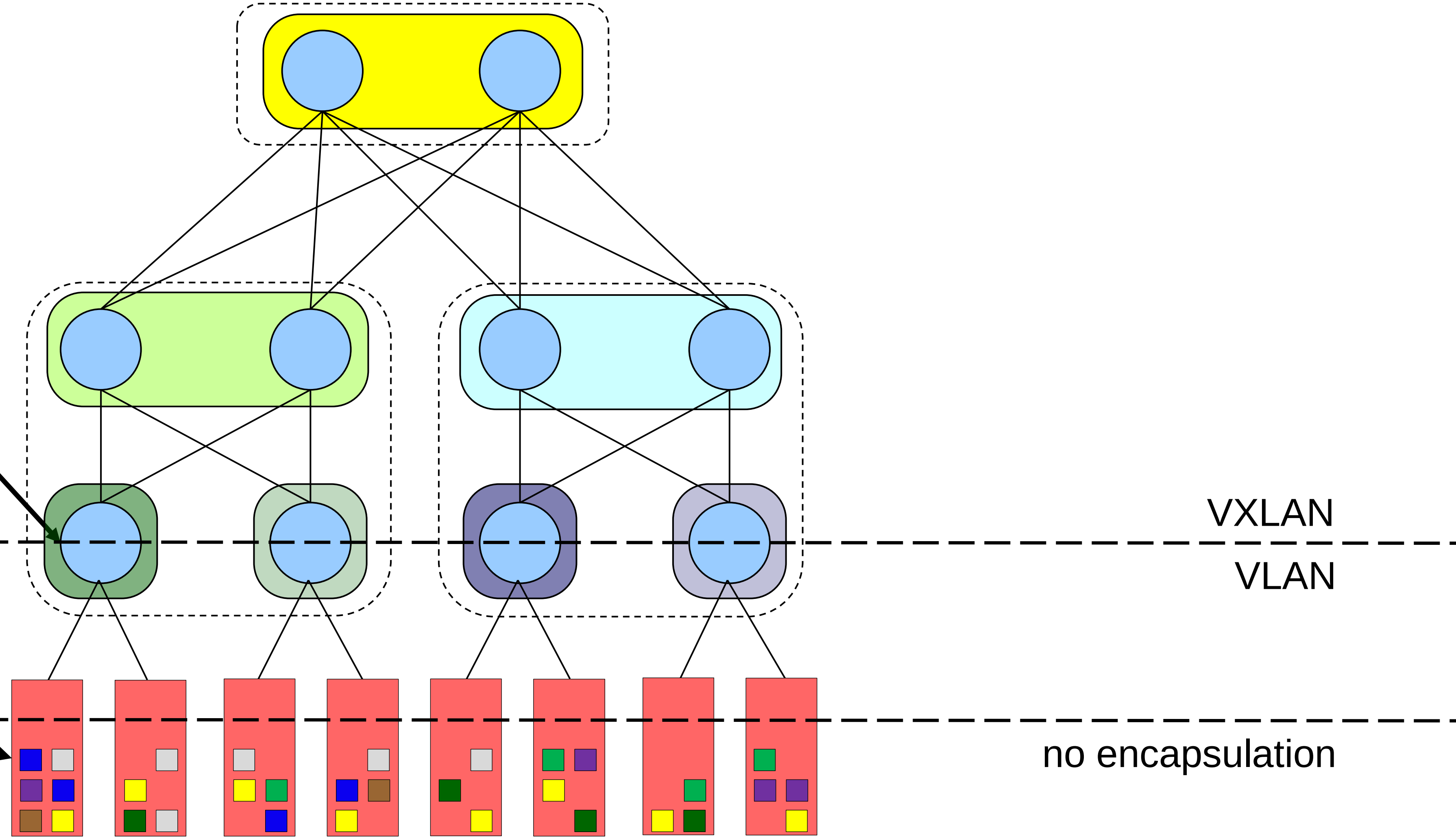
inside the servers

- the server uses the VLAN IDs to forward packets to the correct containers/VMs
- the server untags the packets so that the containers/VMs are unaware of the VLANs
  - containers/VMs of the same tenant share the same virtual Layer-2 network

deploying VXLAN

VLAN ID	VNI
10	5010
20	5020
…	…

VLAN ID	Container
10	1_1_1
20	1_1_3
…	…



the last problem to overcome

- if a Leaf-server link breaks down, the server is severed off the data center
- if a Leaf breaks, all the servers connected to that Leaf are severed off the data center
- if a maintenance needs to be done on a Leaf, all the servers connected to that Leaf are temporarily severed off the data center

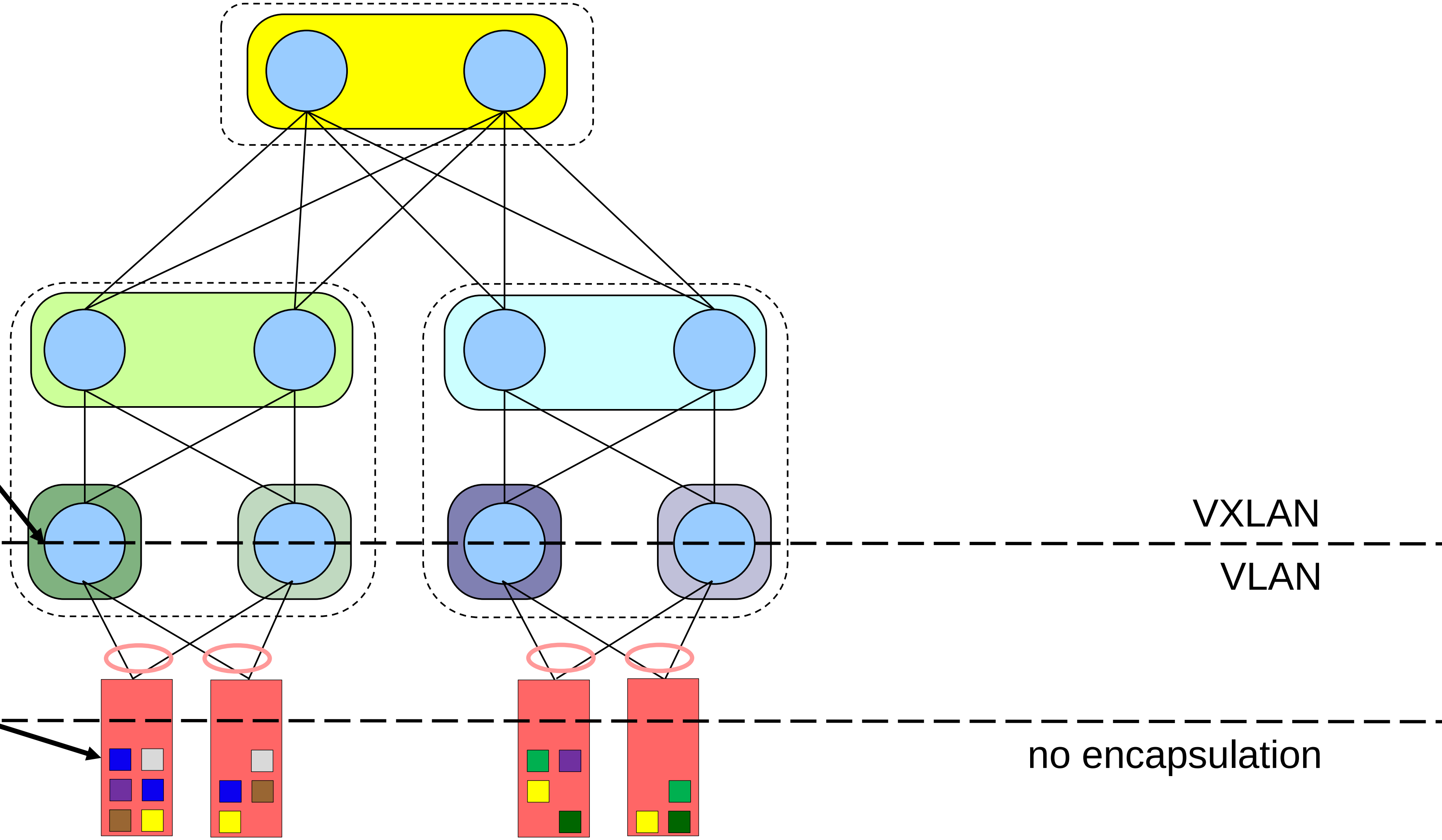
dual attached servers – bonding

- aggregates multiple NICs into a single virtual interface
- Layer-2 technology
- different policies are possible
  - active-backup
  - active-active
    - balance-rr
    - balance-xor
    - 802.3ad
    - and more…

the full picture

VLAN ID	VNI
10	5010
20	5020
…	…

VLAN ID	Container
10	1_1_1
20	1_1_3
…	…



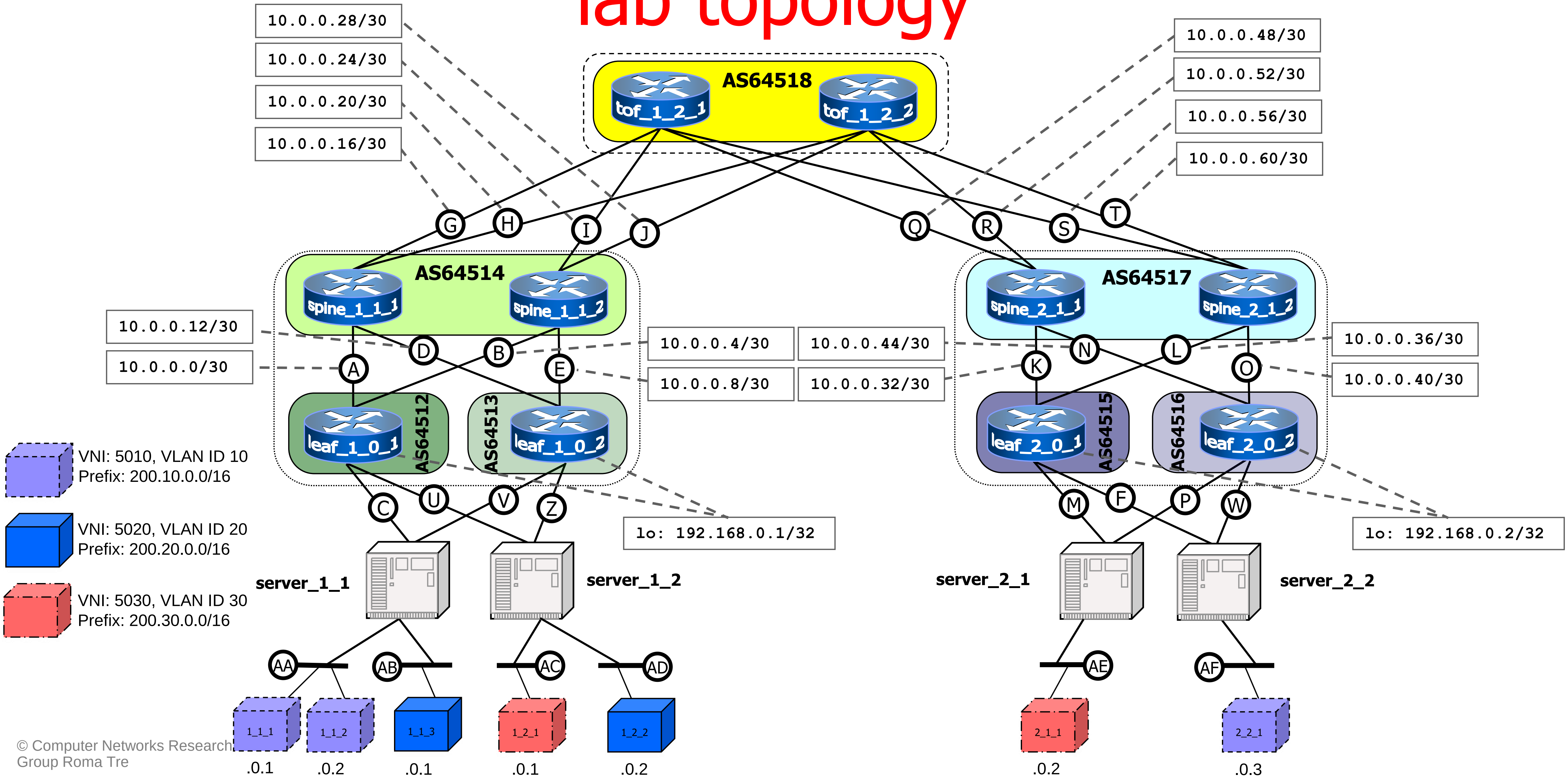
EVPN-BGP Fat-Tree lab

hands on Kathará

anycast BGP

- technique that allows different devices to share the same IP address
- often used in the Internet with DNS servers and CDN servers
- BGP chooses among the nearest instance of the IP address
- in the data center, multipath is exploit to balance over multiple instances of the same anycast IPs

lab topology



leaf configuration example – part 1

leaf_1_0_1.startup

```
ifconfig eth0 10.0.0.1/30 up
ifconfig eth1 10.0.0.5/30 up

# Create loopback
ip addr add 192.168.0.1/32 dev lo
route add 192.168.0.1/32 dev lo

# Setting up VXLAN interfaces
ip link add vtep5010 type vxlan id 5010 dev lo dstport 4789 local 192.168.0.1 nolearning
ip link set up dev vtep5010

ip link add vtep5020 type vxlan id 5020 dev lo dstport 4789 local 192.168.0.1 nolearning
ip link set up dev vtep5020

ip link add vtep5030 type vxlan id 5030 dev lo dstport 4789 local 192.168.0.1 nolearning
ip link set up dev vtep5030
```

configure an anycast IP address on the loopback interface

set source IP for the VTEPs to be the loopback IP

leaf configuration example – part 2

leaf_1_0_1.startup – part 2

```
ip link add bond2 type bond miimon 100 mode 802.3ad
ip link set down dev eth2
ip link set dev eth2 master bond2
ip link set up dev eth2
ip link set up dev bond2

ip link add bond3 type bond miimon 100 mode 802.3ad
ip link set down dev eth3
ip link set dev eth3 master bond3
ip link set up dev eth3
ip link set up dev bond3

# Creating the companion bridge
ip link add br100 type bridge
```

create the bond interface

disable interfaces

connect physical interfaces to the bond

enable the interfaces

leaf configuration example – part 3

leaf_1_0_1.startup – part 3

```
# Attach interfaces to the bridge
ip link set dev vtep5010 master br100
ip link set dev vtep5020 master br100
ip link set dev vtep5030 master br100
ip link set dev bond2 master br100
ip link set dev bond3 master br100

# Enable bridge vlans
ip link set dev br100 type bridge vlan_filtering 1
bridge vlan add vid 10 dev vtep5010 pvid untagged
bridge vlan add vid 20 dev vtep5020 pvid untagged
bridge vlan add vid 30 dev vtep5030 pvid untagged
bridge vlan add vid 10 dev bond2
bridge vlan add vid 20 dev bond2
bridge vlan add vid 20 dev bond3
bridge vlan add vid 30 dev bond3
ip link set up dev br100

/etc/init.d/frr start
```

enable VLANs on the bridge

configure the vtep ports to receive/send untagged traffic of specific VLANs

configure the server ports to receive/send VLAN tagged traffic of specific VLANs

leaf BGP configuration example

bgpd.conf - part 1

```
hostname frr
password frr
enable password frr

router bgp 64512
 timers bgp 3 9
 bgp router-id 192.168.0.1
 no bgp ebgp-requires-policy
 bgp bestpath as-path multipath-relax

neighbor TOR peer-group
 neighbor TOR remote-as external
 neighbor TOR advertisement-interval 0
 neighbor TOR timers connect 10
 neighbor eth0 interface peer-group TOR
 neighbor eth1 interface peer-group TOR
```

enable the l2vpn evpn AFI/SAFI A.F.

bgpd.conf - part 2

```
address-family ipv4 unicast
  neighbor TOR activate
  redistribute connected route-map LOOPBACKS
  maximum-paths 64
exit-address-family

address-family l2vpn evpn
  neighbor TOR activate
  advertise-all-vni
exit-address-family

route-map LOOPBACKS permit 10
  match interface lo
```

route-map to announce the loopback IP

spine BGP configuration example

bgpd.conf - part 1

```
hostname frr
password frr
enable password frr

router bgp 64514
 timers bgp 3 9
 bgp router-id 192.168.0.5
 no bgp ebgp-requires-policy
 bgp bestpath as-path multipath-relax

neighbor TOR peer-group
 neighbor TOR remote-as external
 neighbor TOR advertisement-interval 0
 neighbor TOR timers connect 10
 neighbor eth0 interface peer-group TOR
 neighbor eth1 interface peer-group TOR
```

bgpd.conf - part 2

```
neighbor fabric peer-group
 neighbor fabric remote-as external
 neighbor fabric advertisement-interval 0
 neighbor fabric timers connect 10
 neighbor eth2 interface peer-group fabric
 neighbor eth3 interface peer-group fabric

address-family ipv4 unicast
  neighbor fabric activate
  neighbor TOR activate
  maximum-paths 64
exit-address-family

address-family l2vpn evpn
  neighbor fabric activate
  neighbor TOR activate
exit-address-family
```

activate the l2vpn evpn AFI/SAFI A.F.

ToF BGP configuration example

bgpd.conf - part 1

```
hostname frr
password frr
enable password frr

router bgp 64518
 timers bgp 3 9
 bgp router-id 192.168.0.13
 no bgp ebgp-requires-policy
 bgp bestpath as-path multipath-relax
```

bgpd.conf - part 2

```
neighbor fabric peer-group
 neighbor fabric remote-as external
 neighbor fabric advertisement-interval 0
 neighbor fabric timers connect 10
 neighbor eth0 interface peer-group fabric
 neighbor eth1 interface peer-group fabric
 neighbor eth2 interface peer-group fabric
 neighbor eth3 interface peer-group fabric

address-family ipv4 unicast
  neighbor fabric activate
  maximum-paths 64
exit-address-family

address-family l2vpn evpn
  neighbor fabric activate
exit-address-family
```

activate the l2vpn evpn AFI/SAFI A.F.

server configuration example – part 1

server_1_1.startup

```
ip link add bond1 type bond miimon 100 mode 802.3ad xmit_hash_policy layer3+4 all_slaves_active 1
ip link set dev eth0 down
ip link set dev eth1 down
ip link set dev bond1 down
ip link set eth0 master bond1
ip link set eth1 master bond1
ip link set dev eth0 up
ip link set dev eth1 up
ip link set dev bond1 up

# Creating the bridge
ip link add br100 type bridge

# Attach interfaces to the bridge
ip link set dev bond1 master br100
ip link set dev eth2 master br100
ip link set dev eth3 master br100
```

create the bond interface

disable interfaces

connect physical interfaces to the bond

enable the interfaces

create the bridge

connect the bond and container interfaces to the bridge

why 802.3ad with active-backup?

- usually the right policy is active-active with hash policy
- to support dual attached servers in active-active mode to different switches (leaves), Multi-Chassis Link Aggregation (MLAG) is needed
  - MLAG enables a server or switch with a two-port bond, to connect those ports to different switches and operate as if they are connected to a single, logical switch. This provides greater redundancy and greater system throughput.
- classic Linux kernels do not support MLAG

server configuration example – part 2

server_1_1.startup

```
# Enable bridge vlans
ip link set dev br100 type bridge vlan_filtering 1
bridge vlan add vid 10 dev bond1
bridge vlan add vid 20 dev bond1
bridge vlan add vid 10 dev eth2 pvid untagged
bridge vlan add vid 20 dev eth3 pvid untagged
ip link set up dev br100
```

configure the bond port of the bridge to receive tagged VLAN packets

configure the container ports to send untagged frames

a leaf data plane

```
root@leaf_1_0_1:/# ip route

10.0.0.0/30 dev eth0 proto kernel scope link src 10.0.0.1
10.0.0.4/30 dev eth1 proto kernel scope link src 10.0.0.5
192.168.0.1 dev lo scope link
192.168.0.7 nhid 10 proto bgp metric 20
    nexthop via 10.0.0.2 dev eth0 weight 1
    nexthop via 10.0.0.6 dev eth1 weight 1
```

a leaf BGP control plane

```
leaf_1_1_1# show ip bgp

BGP table version is 2, local router ID is 192.168.0.1, vrf id 0
Default local pref 100, local AS 64512
Status codes:  s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes:  i - IGP, e - EGP, ? - incomplete


   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.0.1/32   0.0.0.0                  0         32768 i
*> 192.168.0.7/32   10.0.0.2                               0 64514 64518 64517 64515 i
*=                  10.0.0.6                               0 64514 64518 64517 64515 i
```

a leaf EVPN control plane

```
leaf_1_1_1# show evpn mac vni 5010

Number of MACs (local and remote) known for this VNI: 3
Flags: N=sync-neighs, I=local-inactive, P=peer-active, X=peer-proxy
MAC               Type   Flags Intf/Remote ES/VTEP            VLAN  Seq #'s
8e:cf:26:1f:44:16 local        eth2                           10    0/0
d2:23:78:4a:e9:02 local        eth2                           10    0/0
76:76:b2:f0:18:6d remote       192.168.0.7                          0/0
```

bibliography and further readings

- [Dutt '18] Dutt, "EVPN in the Data Center", O'Reilly, 2018
- [Bernat '17] Bernat, "VXLAN: BGP EVPN with FRR", https://vincent.bernat.ch/en/blog/2017-vxlan-bgp-evpn
- [Bernat '17] Bernat, "VXLAN & Linux", https://vincent.bernat.ch/en/blog/2017-vxlan-linux
- [RFC-7432] Sajassi, Aggarwal, Bitar, Isaac, Uttaro, Drake, Henderickx, "BGP MPLS-Based Ethernet VPN" Internet Engineering Task Force (IETF) Request for Comments: 7432
- [RFC-8365] Sajassi, Drake, Bitar, Shekhar, Uttaro, Henderickx, "A Network Virtualization Overlay Solution Using Ethernet VPN (EVPN)" Internet Engineering Task Force (IETF) Request for Comments: 8365