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

data centers' architecture and routing

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: Multipath, Fat-Trees, BGP, 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

Multi-Path

in a nutshell

Multi-Path

- data center architectures allow to establish several paths between pairs of nodes
- Multi-Path routing is used to
  - distribute and balance the traffic among different paths
  - increase the fault tolerance

Multi-Path

- allows to have more than one next-hop for the same prefix in the FIB (Forwarding Information Base)
- needs kernel support
  - the main OSes and routers support it
- different usage policies can be applied
- packets of the same *flow* should use the same path
  - reordering packets is computationally heavy

Equal-Cost Multi-Path (ECMP)

- a specific approach to Multi-Path
- exploits paths to the same destination that have the same "cost"
  - e.g., same IGP cost, same number of hops
- allows to choose among the available paths in a uniform way
- often used in Fat-Tree data centers
  - we'll get there in a couple of slides

how packets are forwarded

- for each packet, the kernel decides the FIB entry to be used
- different policies are allowed
  - hash-based policy
    - Layer-3 hash (src IP, dst IP)
    - Layer-4 hash (src IP, dst IP, src port, dst port, protocol)
  - round-robin

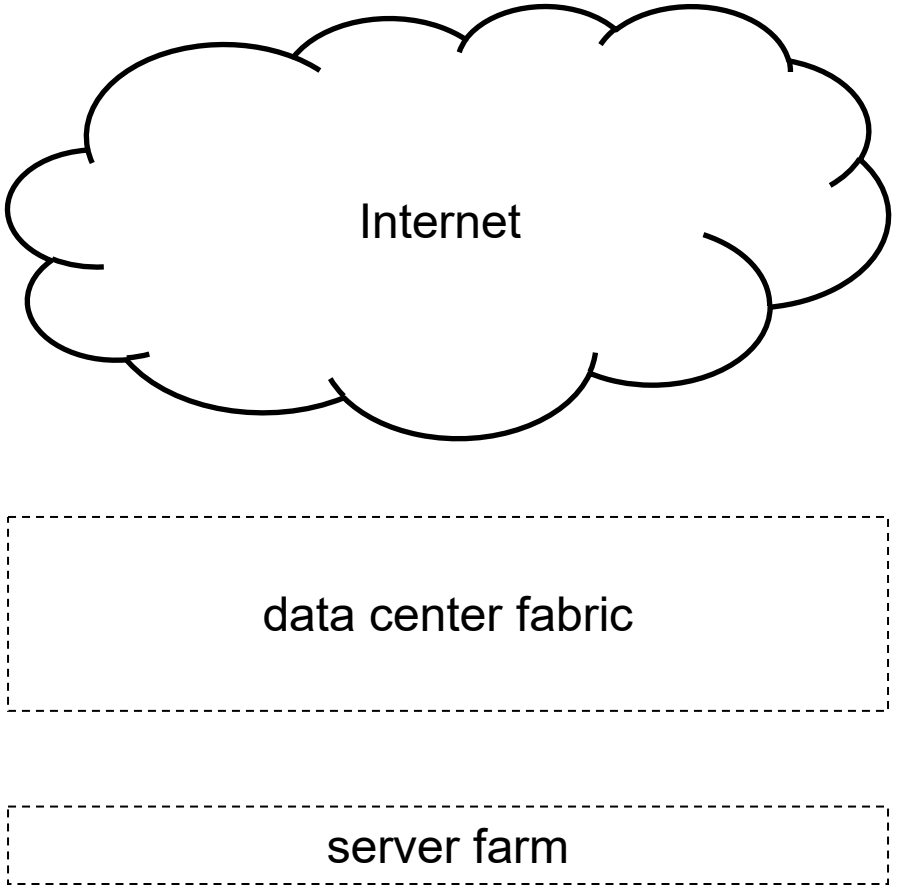
Hyper-scale Data Centers

main concepts

a high-level architecture

- Internet connections
  - multiple redundant high-speed fiber optic links
- fabric
  - infrastructure designed to transport packets between servers and between servers and Internet
- server farm
  - host applications and services

a high-level architecture



fabric architecture

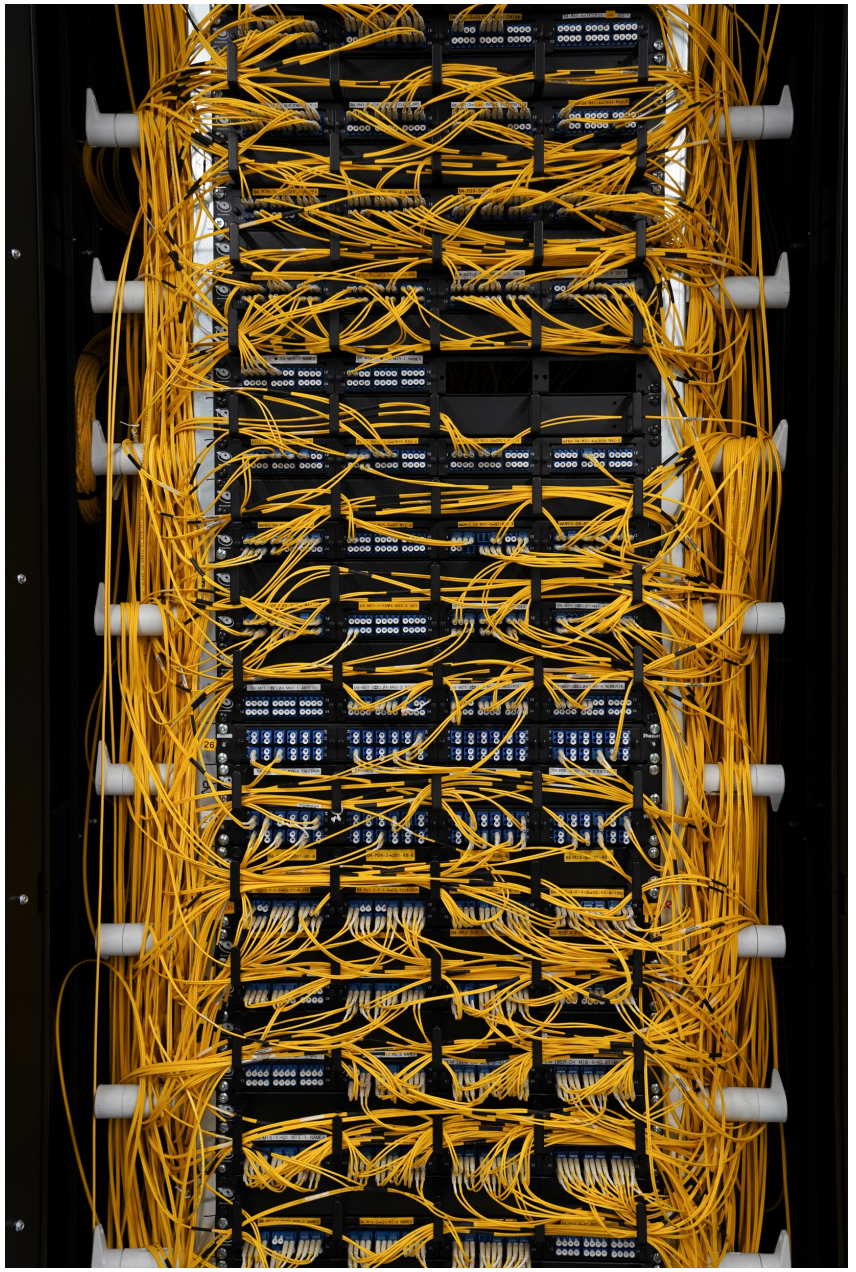
why Fat-Trees?

overview – fabric architecture

- real-life data centers
- service model
- data center network
  - traffic flow directions
  - requirements
  - components
  - topologies

real data center information – figures

- links are fiber optic links
- each link in the fabric is at least at 100Gb/s
- each switch is at least of 64 ports
- about 5,000 switches/routers
- about 60,000 servers



data center failures – yearly report

statistics by Google (2008) in a *portion* of data center composed of 1,800 servers:

- 1,000 individual machine (switch/router or server) failures
- thousands of hard drive failure
- 1 power distribution unit failure
  - down for 500/1000 machines for 6 hours
- 1 cluster complete rewire (not simultaneously)

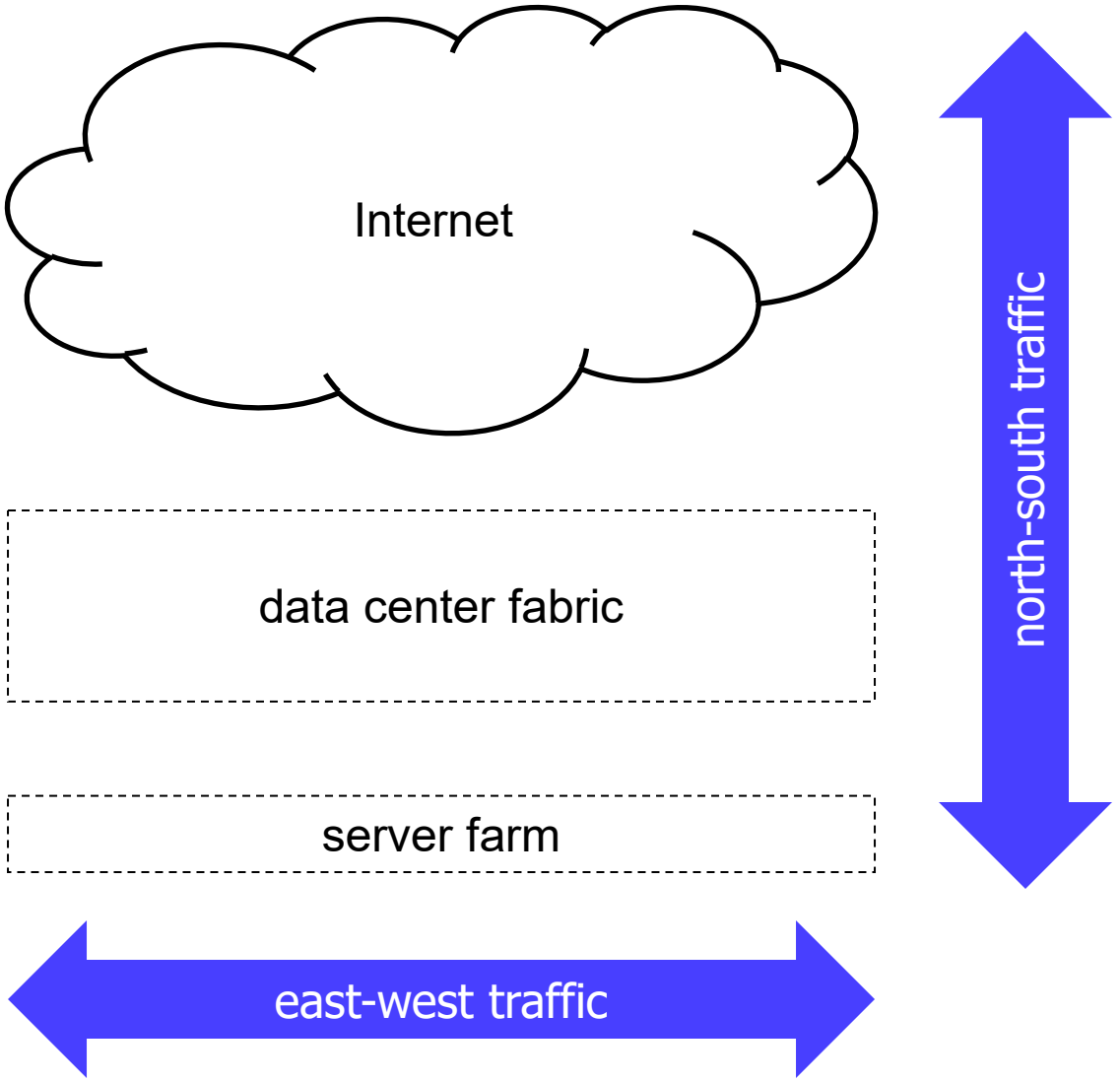
data center service models

- three different service models
  - on-premise data centers
    - built, owned and operated by a company
    - often housed in a building of the organization
  - colocation data centers
    - built and owned by a company that rents space within the data center to other companies
      - the hosting company manages the infrastructure (building, cooling, bandwidth, security etc.)
      - the hosted companies provides the components, including servers, storage, and firewalls
  - cloud data centers
    - applications and data are hosted by a cloud service provider such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP)

traffic flows inside a data center

- two types of flows can be identified
  - traffic exiting or entering the data center
    - also called “north-south traffic”
    - data sent/received via Internet
  - server-to-server communications
    - also called “east-west traffic”: in data center schemata servers are typically drawn side-by-side
    - primarily, supports micro-services and distributed architectures

traffic flows inside a data center



data center network requirements

- support high-bandwidth server-to-server communication
  - applications that rely on cluster computations, such as Hadoop or Spark, can involve hundreds or thousands of servers
  - customer's containers/virtual machines (VMs) are distributed across multiple servers but need to communicate seamlessly
  - microservice architectures heavily rely on server-to-server communication
- scale
  - data centers range from a few hundred to a hundred thousand servers in a single physical location
- resilience
  - data center applications are designed to work in presence of failures

data center network components

- data center nodes can be connected by using two network component types
  - specialized hardware
    - proprietary hardware that can scale to clusters of thousands of nodes with high bandwidth
      - e.g., Google Jupiter and InfiniBand switches
    - expensive option
  - commodity switches and routers
    - cheap option
    - widely adopted, this is the option we consider in the following

topology requirements

- scalability
  - it should be possible to expand the data center
  - in other words, it must be possible to buy and deploy hardware similar to the one already deployed
- bandwidth
  - hosts in the fabric should communicate with each other using the full bandwidth of their NICs

data center network topologies

- several topologies have been proposed
  - Clos
  - Fat-Tree
  - VL2
  - Jellyfish
  - Xpander
  - DCell
  - ...

Clos topology

- invented by Edson Erwin in 1938 to address scalability issues in telephone networking
  - formalized by Charles Clos in 1953
- the original problem was allowing N contemporary connections from N input lines to N output lines without using a single crossbar switch

Fat-Tree topology

- Fat-Trees were originally introduced by Charles Leiserson in 1985
- the Fat-Tree is a special case of a Clos
- typical Fat-Tree architectures today consist of three level of nodes (switches/routers)
- high redundancy
- constant *bisection* of the available bandwidth
- typically, nodes have the same characteristics
  - easier to stock spare equipments

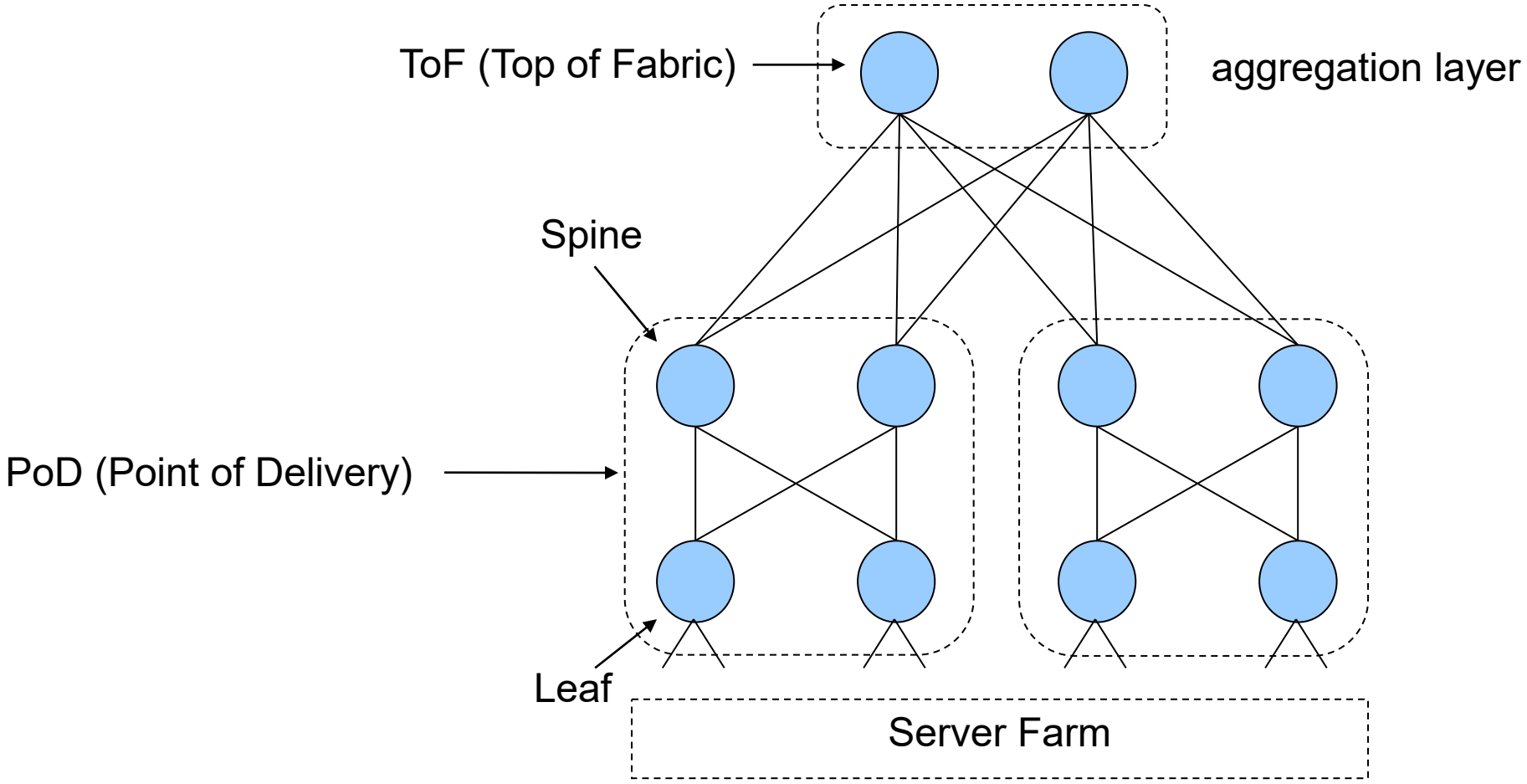
Fat-Tree parameters

- the Fat-Tree is a modular topology
- two parameters define a Fat-Tree
  - *radix* of the nodes (even number, denoted by 2K)
    - number of available ports
  - *redundancy factor* (denoted by R)
- usually, if the radix of a node is 2K it has K connections towards the north and K connections towards the south
  - different choices are possible, but not considered here

Fat-Tree nodes

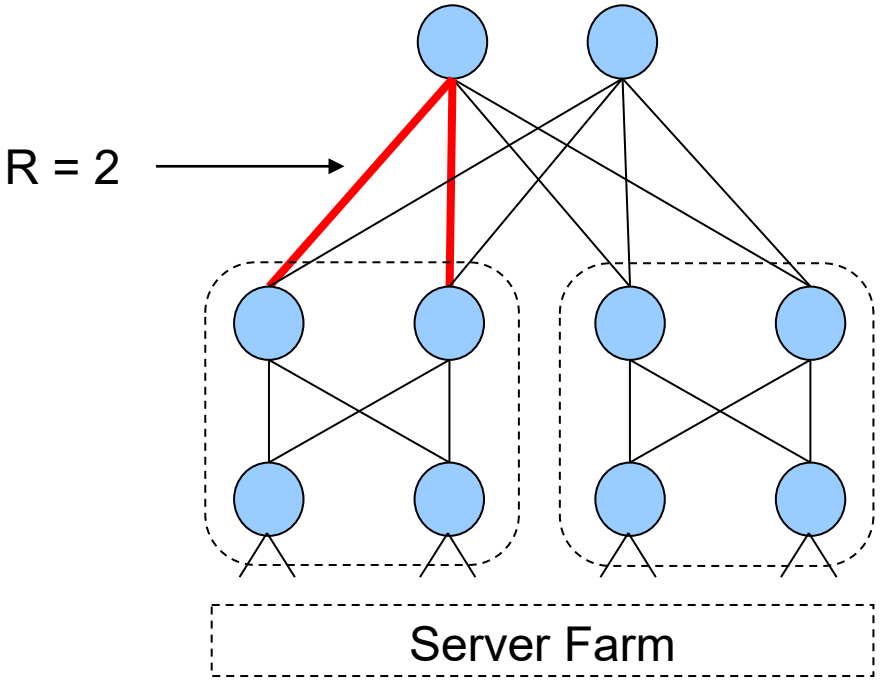
- Leaf
  - node connected to the server farm
- Spine
  - node north of Leaves and south of ToF nodes
- Point of Delivery (PoD)
  - set of fully interconnected Leaves and Spines
- Top of Fabric (ToF)
  - set of top nodes that provide inter-PoD communication

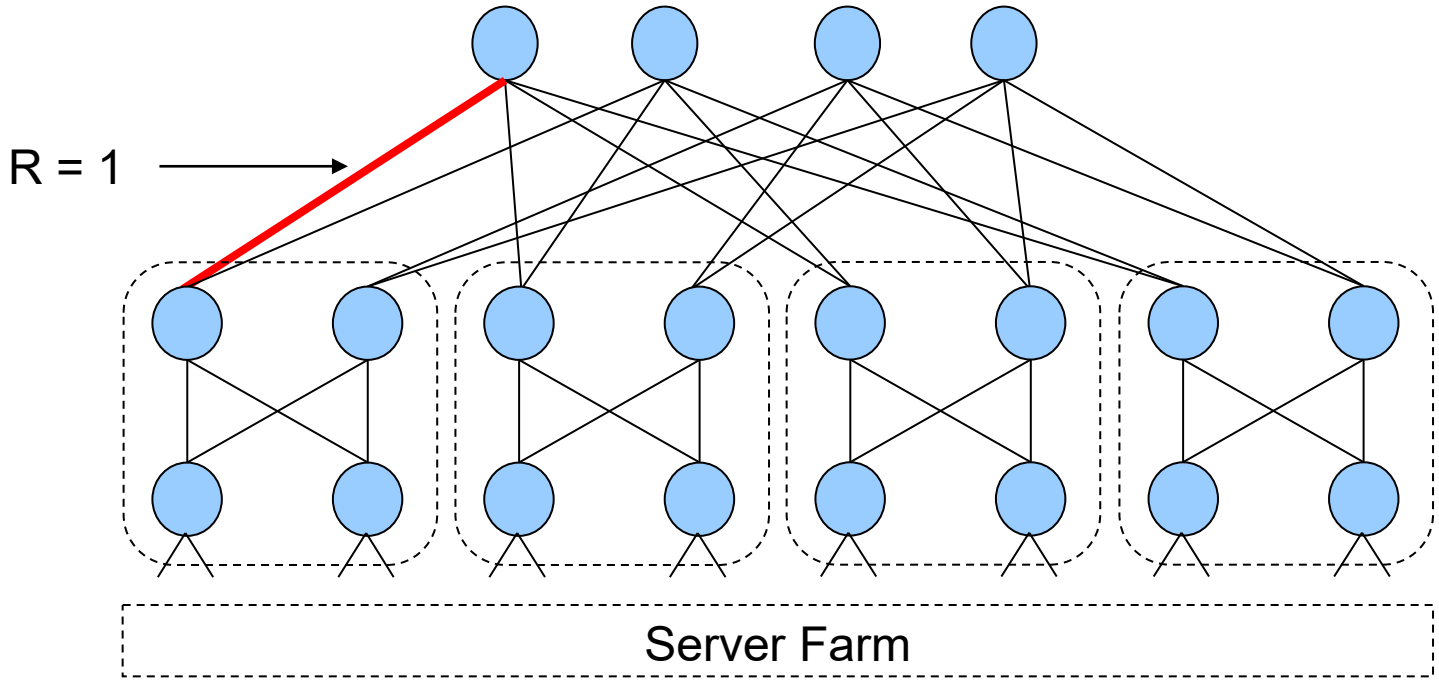
an example of Fat-Tree: FT(K=2,R=2)



redundancy factor (R)

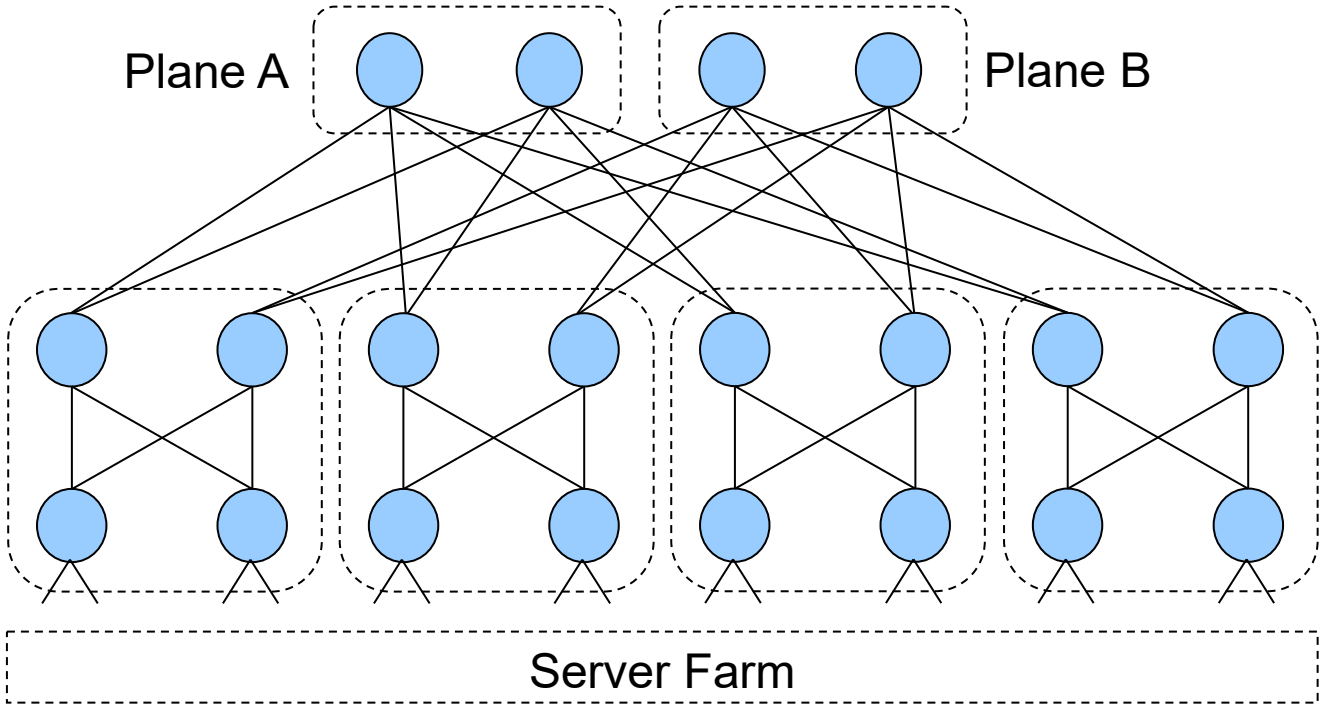
- number of links between a ToF node and a PoD
- allow to connect more PoDs reducing the redundancy





multi-plane Fat-Tree

- when K≠R the Fat-Tree is called *multi-plane*
- the ToF nodes are partitioned in sets called *planes*



example of a multi-plane Fat-Tree

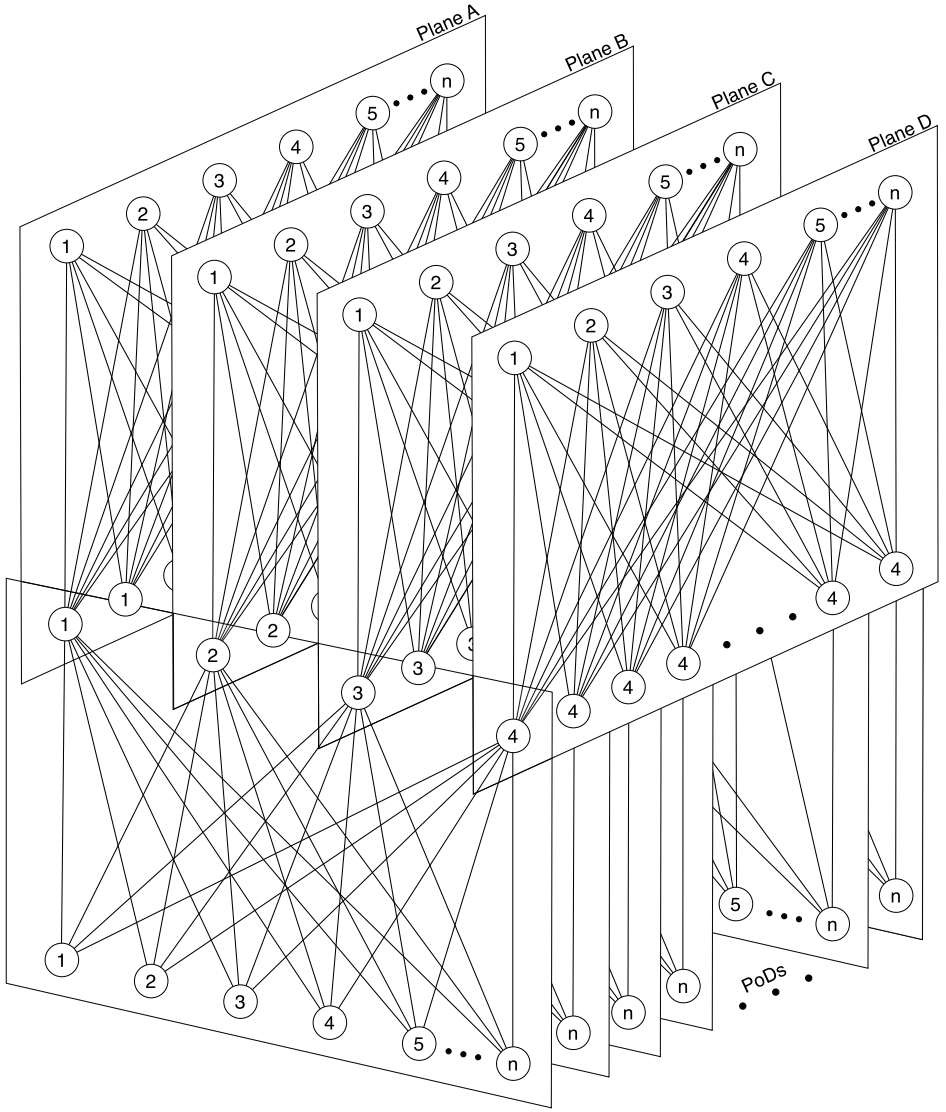
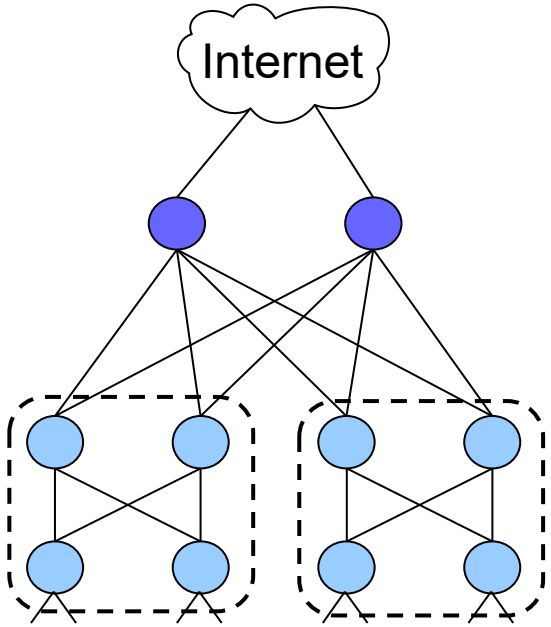
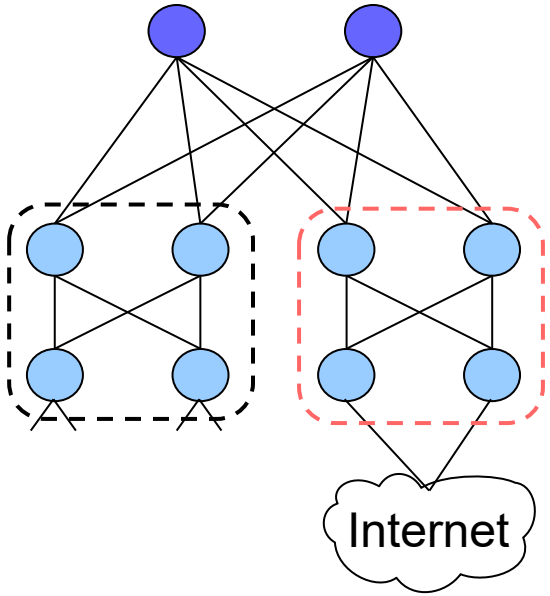


diagram from the
IETF RIFT Draft

connecting a Fat-Tree to the Internet

- two strategies
  - usage of a dedicated PoD
  - usage of ToFs





routing in Fat-Trees

why BGP?

overview – routing

- why routing and not switching?
- the choice of the routing protocols
- why using BGP in Fat-Tree data centers?
- inter-domain routing VS data center routing
- BGP in the data center

why routing and not switching?

	Pros	Cons
Switching (L2)	No configuration needed	• STP protocol has no multipath support • Using VLANs for load balancing is tricky • Broadcast traffic flooding the network
Routing (L3)	Multipath support	• Need of routing protocols • Complex configuration • Need of automating the configuration

choosing a routing protocol

- there are many possibilities, among them
  - a classical IGP protocol (e.g., OSPF, IS-IS)
  - BGP (most used in Hyperscale data centers)
  - RIFT (Routing In Fat-Trees)
    - Under standardization at IETF
    - Designed for Fat-Trees topologies
  - OpenFabric
    - IS-IS variant created for Clos topologies
  - SDN Protocols

why using BGP in Fat-Tree data centers?

- BGP has several stable and robust implementations, even open-source
  - e.g., FRRouting, Quagga
- BGP generates less flooding than common IGPs (IS-IS, OSPF, etc.)
  - if a received update does not change the best route, a BGP speaker does not propagate the update
- BGP natively supports ECMP (Equal-Cost Multi-Path)
  - Fat-Trees have many paths with the same length

why using eBGP? (and not iBGP)

- the most obvious choice would be to use iBGP since the data center networks is under the same administration
- however, eBGP is always used because
  - eBGP is easier to setup, no need for IGP
    - with iBGP, the IGP would compute routes
  - iBGP multipath support has some limitations
    - can be overcame, but it is complex

inter-domain routing VS DC routing

inter-domain routing	**data centers' routing**
the internet has relatively sparse connectivity	data centers' networks have very dense connectivity
stability is preferred over quick convergence	quick convergence is preferred over stability
the aim is computing a single best path for each destination	the aim is computing multiple paths to each destination

BGP in the data center

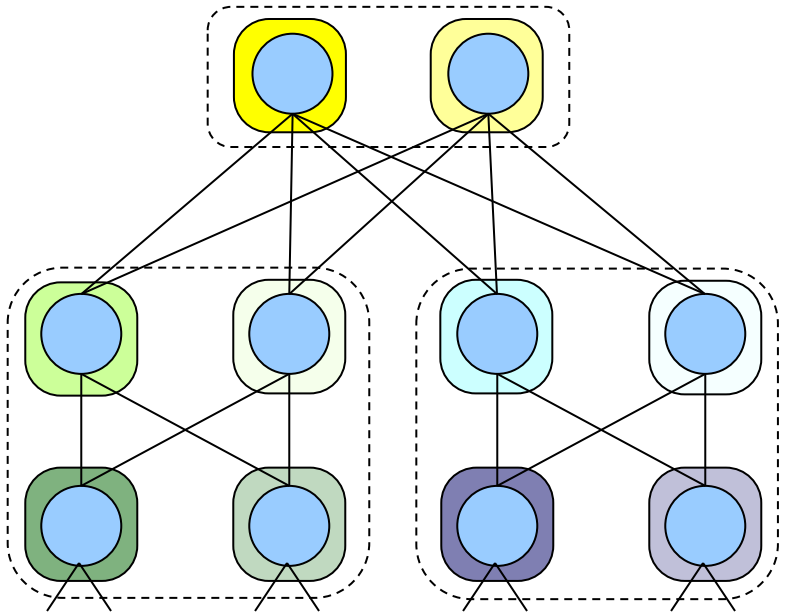
- BGP was primarily devised for inter-domain routing
  - default configurations are not suitable in a data center
- needs some tweaks described in RFC-7938
  - AS numbers assignment
  - ECMP *policy relax*
  - timer adjustment

AS numbers assignment

- global ASNs are not used in the data center
  - they can be misleading
    - because operators associate them with names
  - they are dangerous
    - an accidental leakage of the internal BGP notifications to the internet may be disruptive
- private ASNs are generally used
  - the 2-byte ASNs allow only 1,023 private ASes in the range 64512–65534
  - the 4-byte ASNs support almost 95 million private ASes in the range 4,200,000,000–4,294,967,294

ASes and routers

- the most obvious choice would be assigning a different ASN to each node
- however, this approach would lead to BGP *path exploration* issues

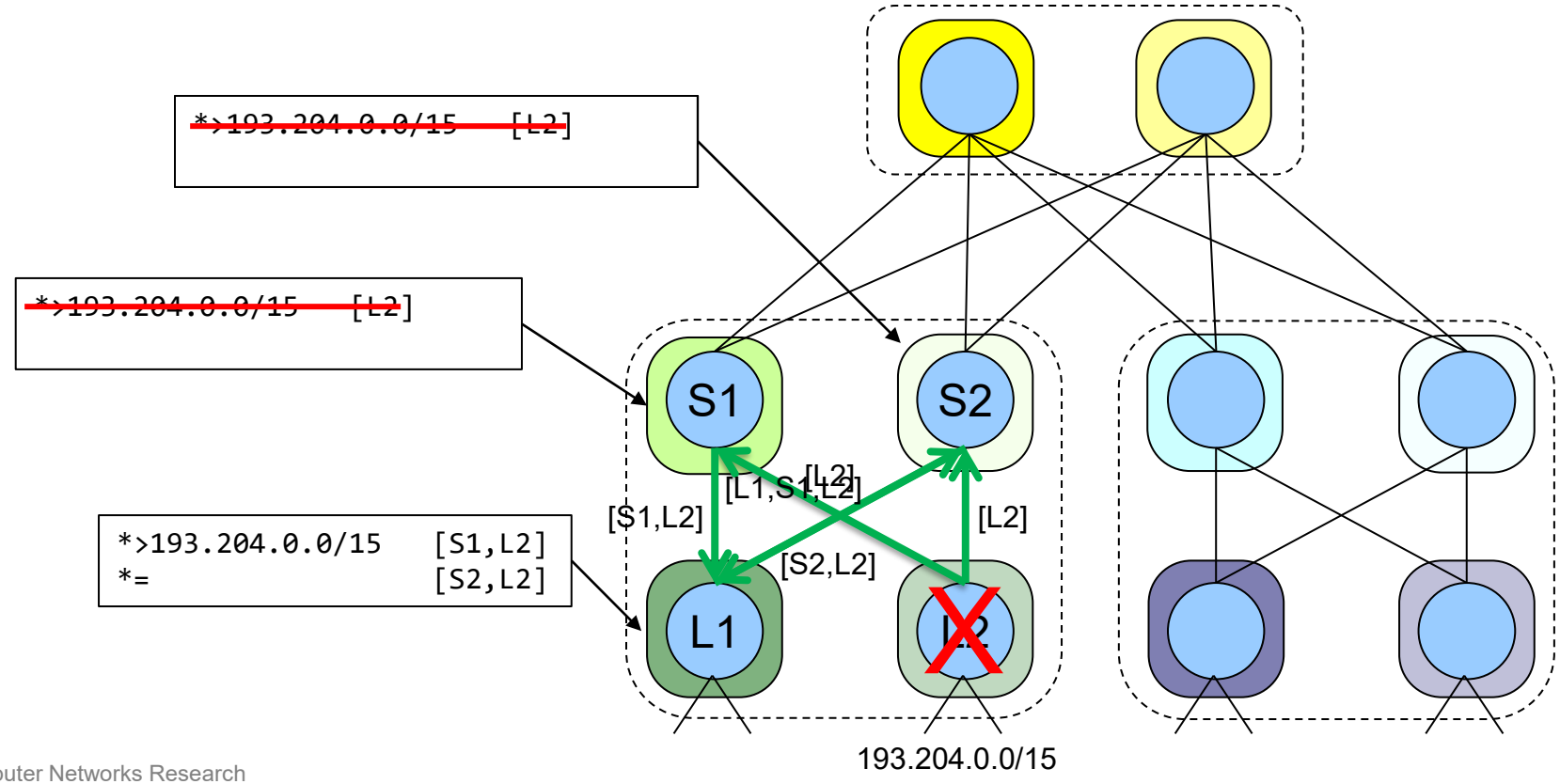


BGP path exploration

- immediately after a fault there is a transient period when inconsistent AS-paths are propagated in the network
  - routers have plenty of alternatives and jump from one to another before all alternatives are withdrawn
  - each best route change is propagated again by the routers
  - lots of useless BGP updates are transmitted
- since data centers' topologies are dense and hence have a lot of cycles this problem has to be addressed

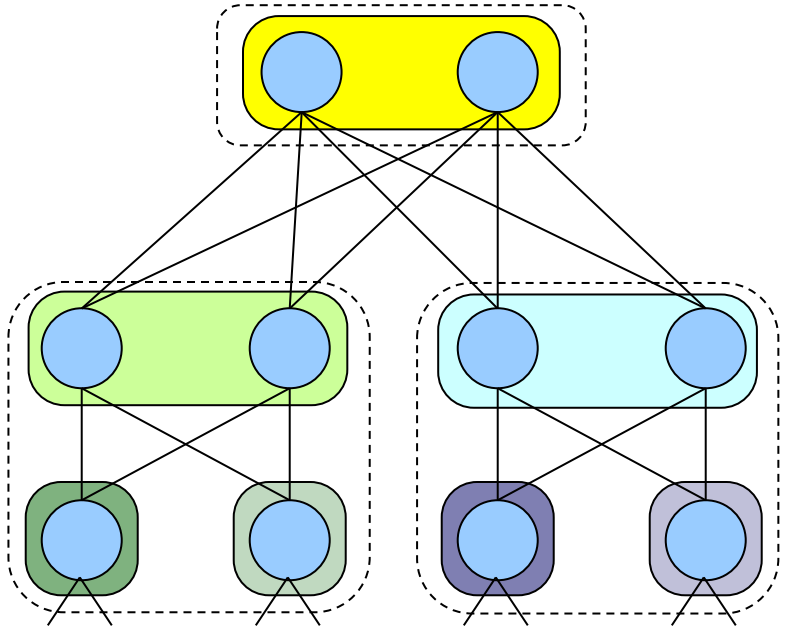
the problem of BGP path exploration

- because of the adopted AS scheme several fake routes could be possible during path exploration



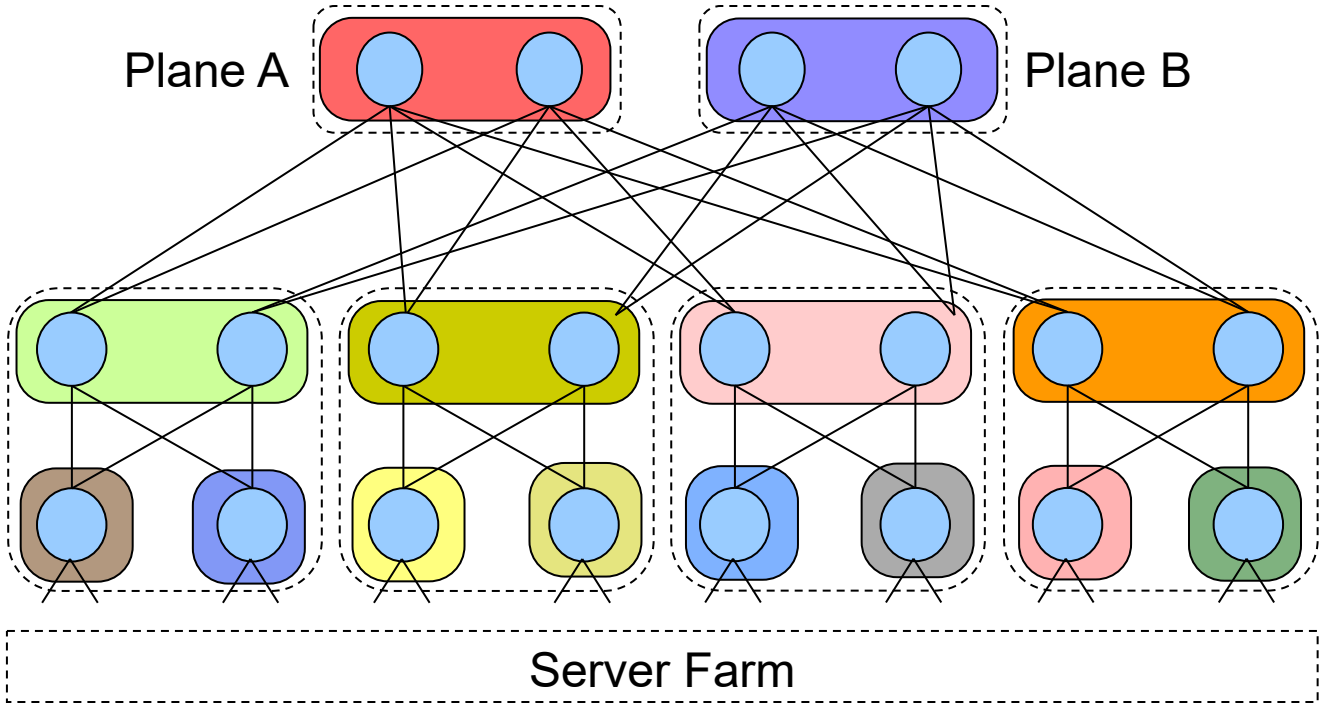
the ASes scheme of choice

- each Leaf is a different AS
- the Spine nodes of each PoD belong to the same AS
- the ToF nodes of the same plane belong to the same AS



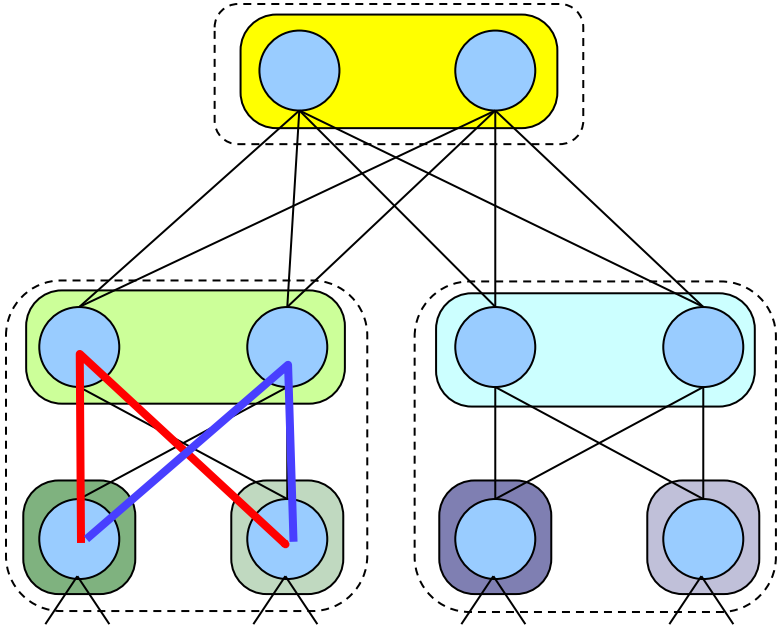
BEWARE: there are NO iBGP peerings

the ASes scheme of choice – multi-plane



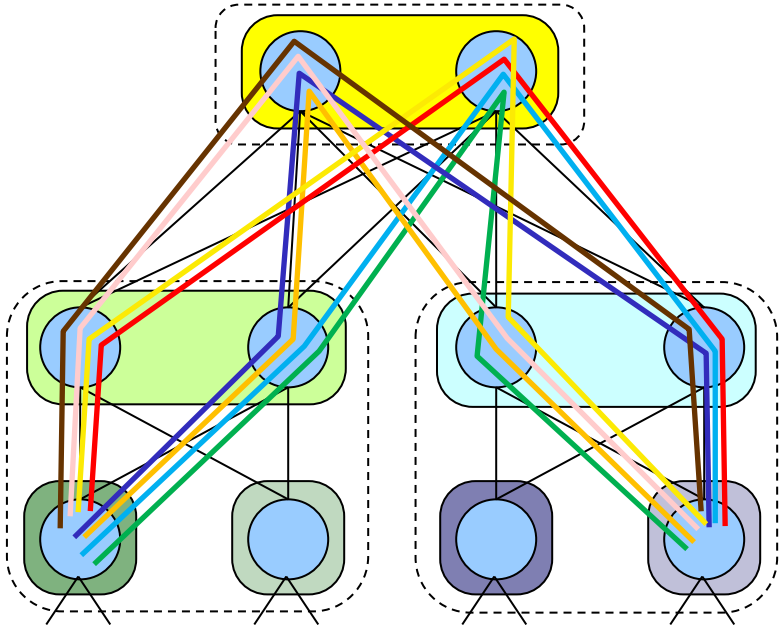
multiple paths between Leaves

- the adopted AS scheme allows for multiple paths between pairs of Leaves
  - in a FT(K=2,R=2) only two paths are possible between two Leaves of the same PoD
    - they don’t leave the PoD



multiple paths between Leaves

- the adopted AS scheme allows for multiple paths between pairs of Leaves
  - in a FT(K=2,R=2) eight paths are possible between two Leaves of different PoDs



BGP and ECMP

- BGP natively supports ECMP
- by default, BGP considers two announcement of the same prefix "equal" if they are equal in each best-path selection criterion except for the last one
  - lowest router-id of the announcing peer
- to enable multi-path, BGP requires that the AS-paths selected for multi-path match exactly
  - not just they have equal-length but equal AS numbers inside

BGP multi-path relax

- when the AS_PATH lengths of different announcements for the same prefix are the same, the best-path algorithm skips checking for exact match of the AS numbers
- this modification is often called “as-path multipath-relax”
  - different vendors may use different names
- really needed for using dual attached servers and multi-plane Fat-Trees

tuning BGP timers

- there are four main timers that are responsible for BGP behaviour
  - advertisement interval timer
  - keepalive timer
  - hold timer
  - connect timer

advertisement interval timer

- announcements that need to be sent to a neighbor are bunched together and sent together only when the interval expires
  - then the timer is reset for that neighbor
- the default value for eBGP is 30s
  - in interdomain routing this improves stability and reduces the number of multiple updates for the same prefix
- in data centers is set to 0s
  - it is required for fast convergence

keepalive and hold timers

- each BGP peer sends periodic keepalive messages to its neighbors according to the keepalive timer
- when a peer doesn’t receive a keepalive for a period greater than the hold timer
  - the connection is dropped
  - all the announcements received are considered invalid
  - the peer tries to re-establish the connection
- by default, the keepalive timer is 60s and the hold timer is 180s
- in data centers timers of 3s and 9s are used, respectively

connect timer

- when a connection to a peer fails, BGP waits for the connect timer expiration before attempting to reconnect
  - the connect timer by default is 60s
  - this can delay session re-establishment when a link recovers from a failure or a node powers up
- in data centers it is set to 10s

automating the configuration

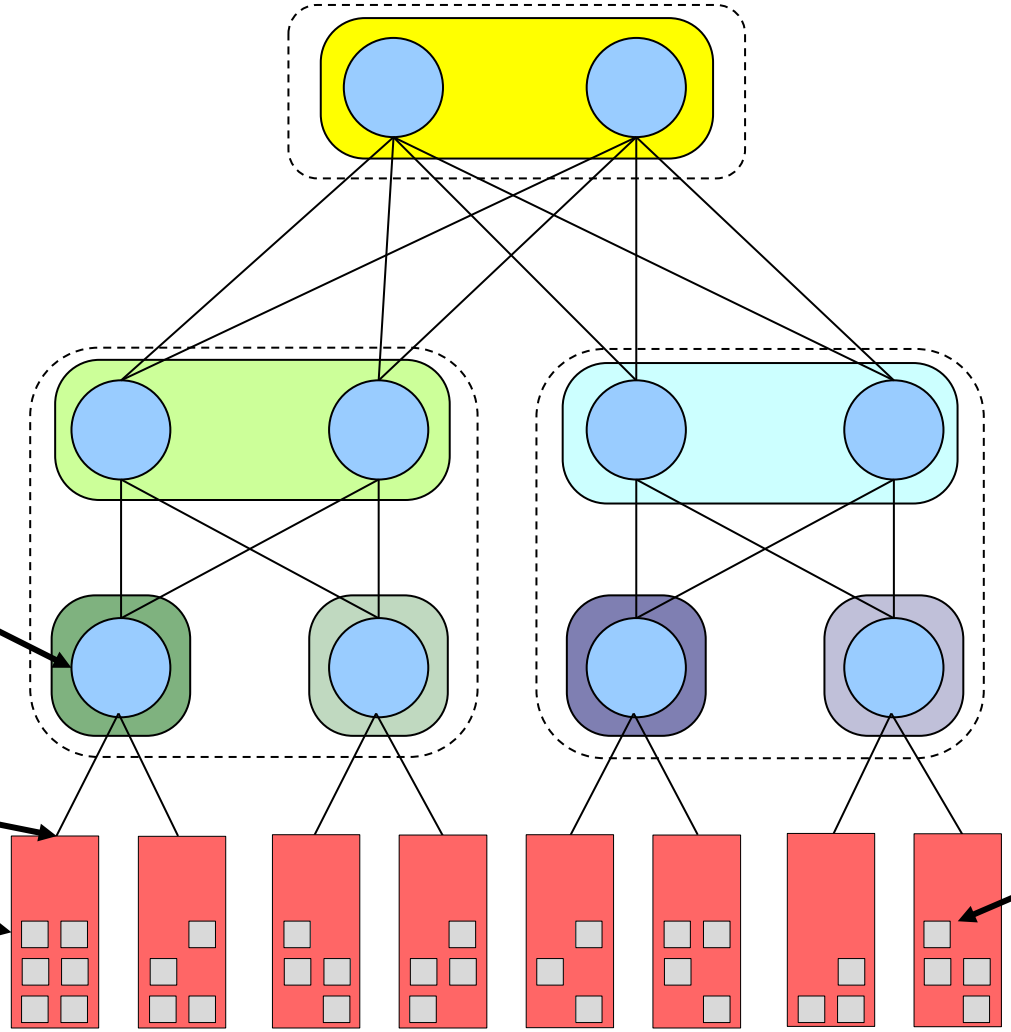
- unnumbered interfaces
  - used to establish peerings specifying the interface name rather than the IP address and the remote AS number
- peer groups
  - used to specify policies for groups of peers

connecting the servers

each Leaf announces its server's prefix

IPs are assigned directly to the servers

Servers



Containers or VMs

disadvantages

- containers/VMs share the same IP prefix of the server
  - no possibility to move containers between servers without IP remapping
- tenants must follow the IP plan of the data center
  - cannot expose containers with custom IPs
- there is no isolation between
  - the data center traffic and the tenants traffic
  - different tenants

basic Fat-Tree lab

hands on Kathará

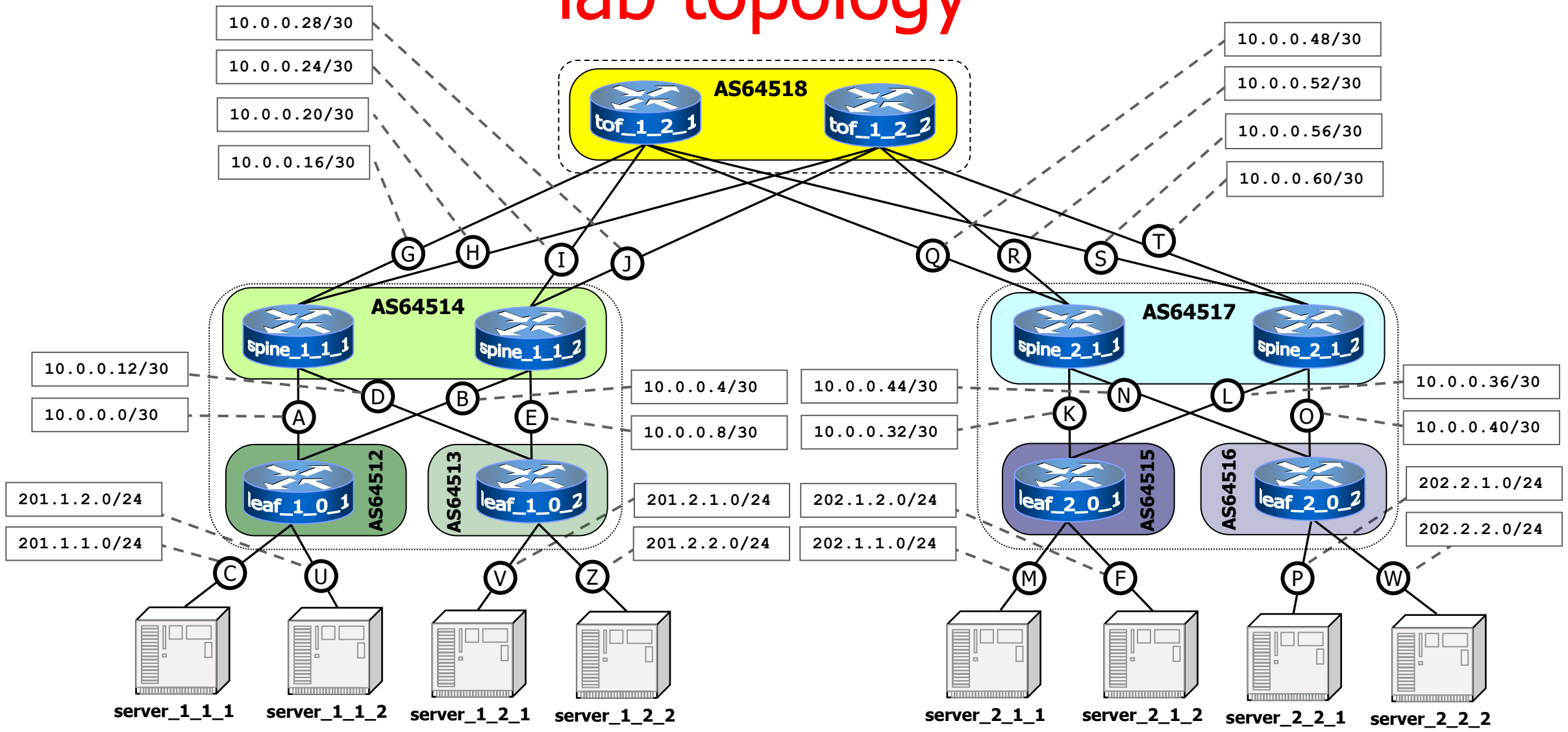
lab pre-conditions

- Linux and macOS
  - no specific requirement
- Windows
  - WSL 2 does not support Multi-Path
  - need to fallback to Hyper-V Docker backend
    - open Docker Desktop and go to Settings (cog in the top-right corner)
    - unselect the “Use the WSL 2 based engine” checkbox
    - click “Apply & restart”
  - or create a VM with a Linux distribution

naming convention

- tof_x_y_z
  - x: plane number
  - y: level, always 2
  - z: ToF number
- spine_x_y_z
  - x: PoD number
  - y: level, always 1
  - z: Spine number
- leaf_x_y_z
  - x: PoD number
  - y: level, always 0
  - z: Leaf number
- server_x_y_z
  - x: PoD number
  - y: corresponding Leaf number
  - z: server number

lab topology



ToF configuration example

bgpd.conf

```
router bgp 64518
 timers bgp 3 9
 bgp router-id 192.168.0.14
 no bgp ebgp-requires-policy
 bgp bestpath as-path multipath-relax

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
```

- set keepalive and hold timers
- BGP multi-path relax
- create a peer-group
- all peers of the group are eBGP. The remote ASN is inferred
- set advertisement interval timer
- set connect timer
- add interfaces to peer-group for unnumbered peering
- enable A.F. to peer-group
- maximum Multi-Path possibilities

Spine configuration example

bgpd.conf – part 1

```
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

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
```

bgpd.conf – part 2

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

Leaf configuration example

bgpd.conf

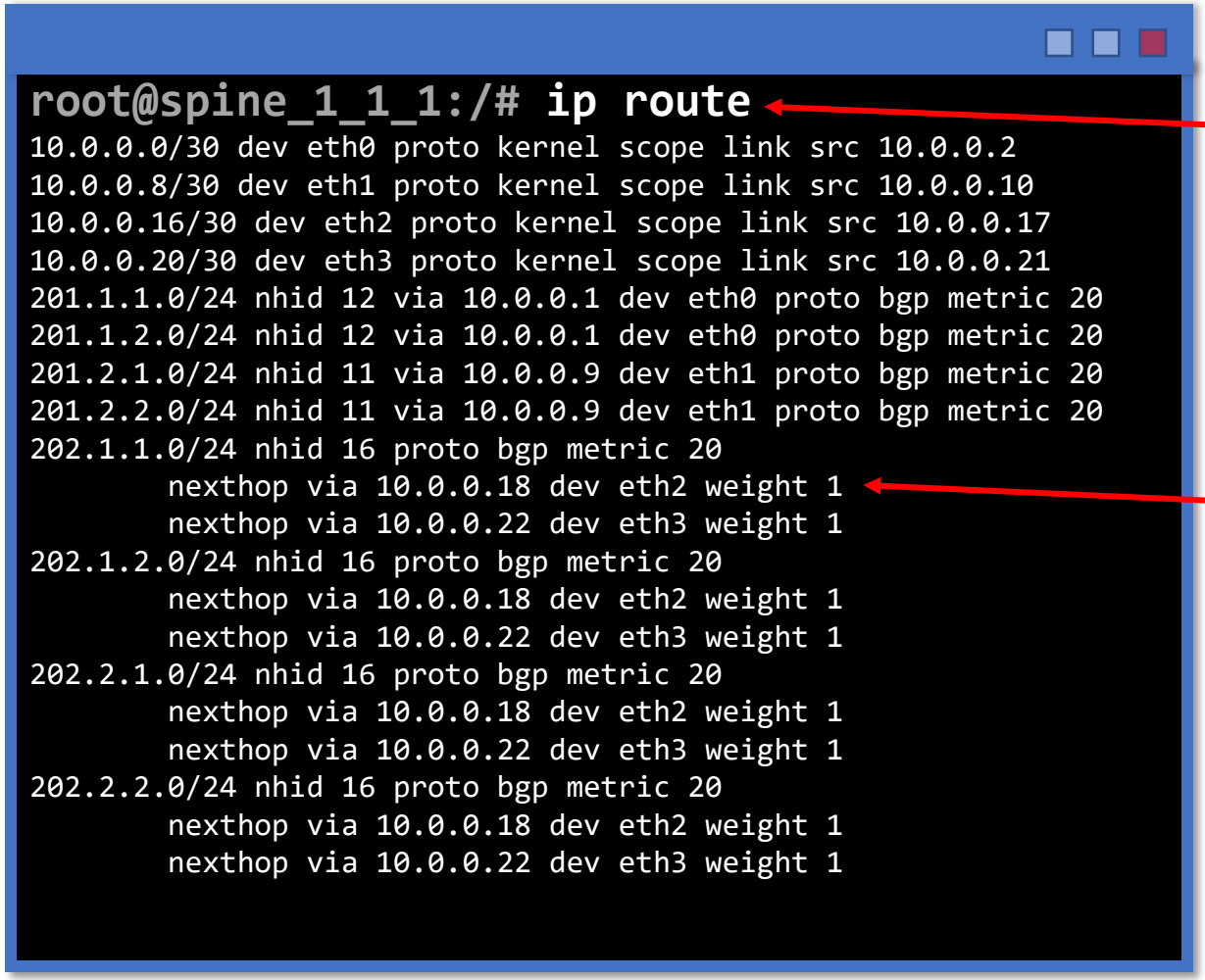
```
router bgp 64512
 timers bgp 3 9
 bgp router-id 192.168.0.1
 no bgp ebgp-requires-policy
 bgp bestpath as-path multipath-relax

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

address-family ipv4 unicast
  neighbor TOR activate
  network 201.1.1.0/24
  network 201.1.2.0/24
  maximum-paths 64
exit-address-family
```

data plane

unable to view multi-path routes

```
root@spine_1_1_1:/# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.4        0.0.0.0         255.255.255.252 U     0      0        0 eth0
10.0.0.12       0.0.0.0         255.255.255.252 U     0      0        0 eth1
10.0.0.24       0.0.0.0         255.255.255.252 U     0      0        0 eth2
10.0.0.28       0.0.0.0         255.255.255.252 U     0      0        0 eth3
201.1.1.0       10.0.0.5        255.255.255.0   UG    20     0        0 eth0
201.1.2.0       10.0.0.5        255.255.255.0   UG    20     0        0 eth0
201.2.1.0       10.0.0.13       255.255.255.0   UG    20     0        0 eth1
201.2.2.0       10.0.0.13       255.255.255.0   UG    20     0        0 eth1
202.1.1.0       10.0.0.26       255.255.255.0   UG    20     0        0 eth2
202.1.2.0       10.0.0.26       255.255.255.0   UG    20     0        0 eth2
202.2.1.0       10.0.0.26       255.255.255.0   UG    20     0        0 eth2
202.2.2.0       10.0.0.26       255.255.255.0   UG    20     0        0 eth2
```

data plane



command to manage routing tables

multiple next-hop for the same prefix

control plane

```
spine_1_1_1# show ip bgp
BGP table version is 8, local router ID is 192.168.0.6, vrf id 0
Default local pref 100, local AS 64514
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
*> 201.1.1.0/24     10.0.0.5                 0             0 64512 i
*> 201.1.2.0/24     10.0.0.5                 0             0 64512 i
*> 201.2.1.0/24     10.0.0.13                0             0 64513 i
*> 201.2.2.0/24     10.0.0.13                0             0 64513 i
*> 202.1.1.0/24     10.0.0.26                              0 64518 64517 64515 i
*=                  10.0.0.30                              0 64518 64517 64515 i
*> 202.1.2.0/24     10.0.0.26                              0 64518 64517 64515 i
*=                  10.0.0.30                              0 64518 64517 64515 i
*> 202.2.1.0/24     10.0.0.26                              0 64518 64517 64516 i
*=                  10.0.0.30                              0 64518 64517 64516 i
*> 202.2.2.0/24     10.0.0.26                              0 64518 64517 64516 i
*=                  10.0.0.30                              0 64518 64517 64516 i

Displayed  8 routes and 12 total paths
```

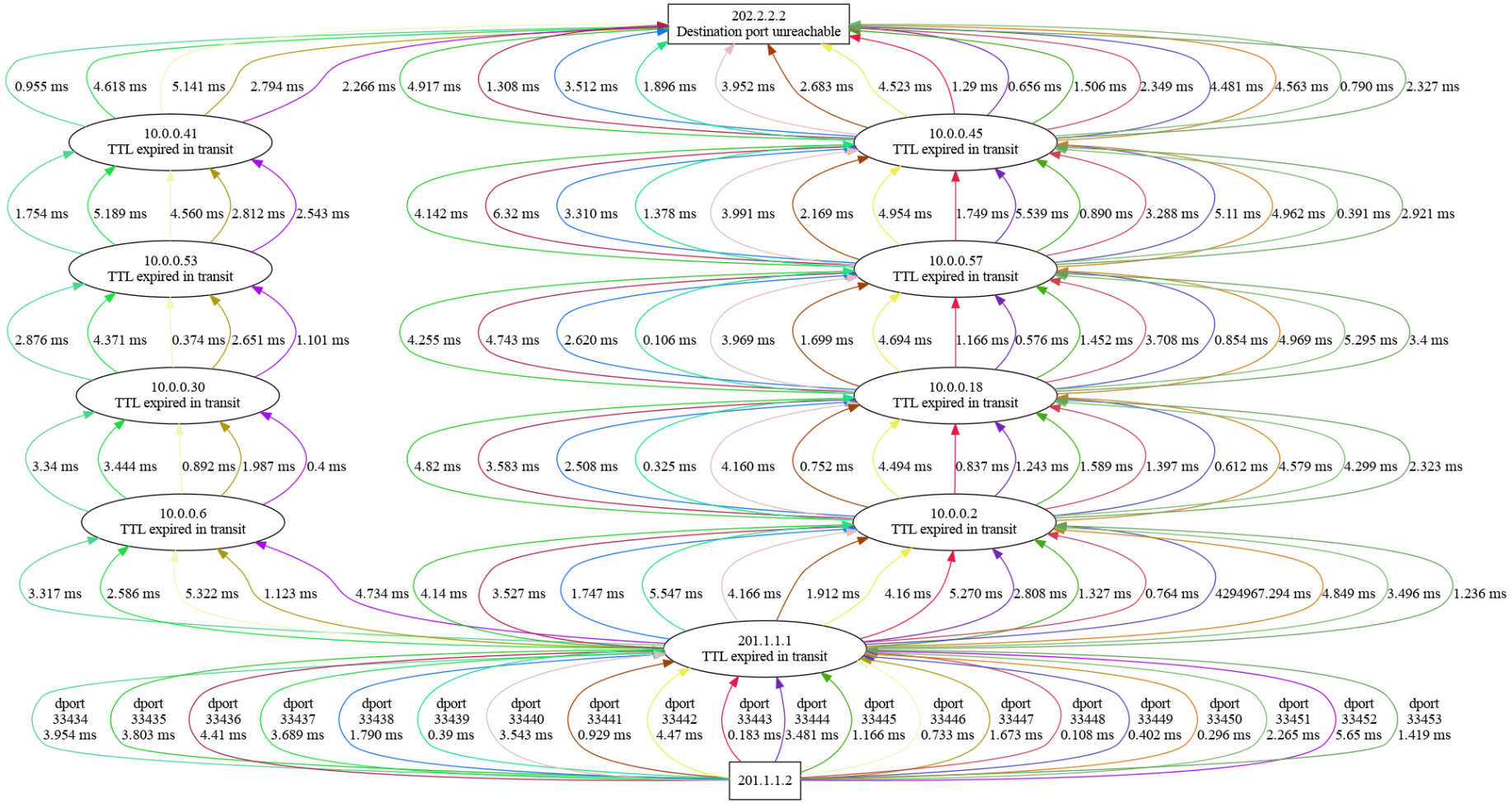
Multi-Path traceroute

- traditional traceroute may provide hard-to-interpret or even misleading results when used in presence of ECMP
- Multi-Path traceroute tools generate packet header contents to obtain a more precise picture of the actual routes of packets
  - allow all probes towards a destination to follow the same path in the presence of per-flow load balancing
  - allow a user to distinguish between the presence of per-flow load balancing and per-packet load balancing

Multi-Path traceroute

- two different tools:
  - `paris-traceroute`
    - traceroute designed to work in presence of Multi-Path and load balancers
  - `dublin-traceroute`
    - based on the `paris-traceroute`
    - adds a NAT detection technique
    - introduces visualization and analysis tools

dublin-traceroute example output



bibliography and further readings

- [Caiazzi ‘22] Caiazzi, Scazzariello, Alberro, Ariemma, Castro, Grampin, Di Battista, “Sibyl: a Framework for Evaluating the Implementation of Routing Protocols in Fat-Trees“, NOMS 2022
- [Caiazzi ‘21] Caiazzi, Scazzariello, Ariemma, “VFTGen: a Tool to Perform Experiments in Virtual Fat Tree Topologies“, IM 2021
- [Caiazzi ‘19] Caiazzi, “Software Defined Data Centers: methods and tools for routing protocol verification and comparison”, Ms. Thesis, Roma Tre University, 2019
- [Dutt ‘17] Dutt, “BGP in the Data Center”, O’Reilly, 2017
- [RFC-7938] Lapukhov, Premji, “Use of BGP for Routing in Large-Scale Data Centers” Internet Engineering Task Force (IETF) Request for Comments: 7938

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

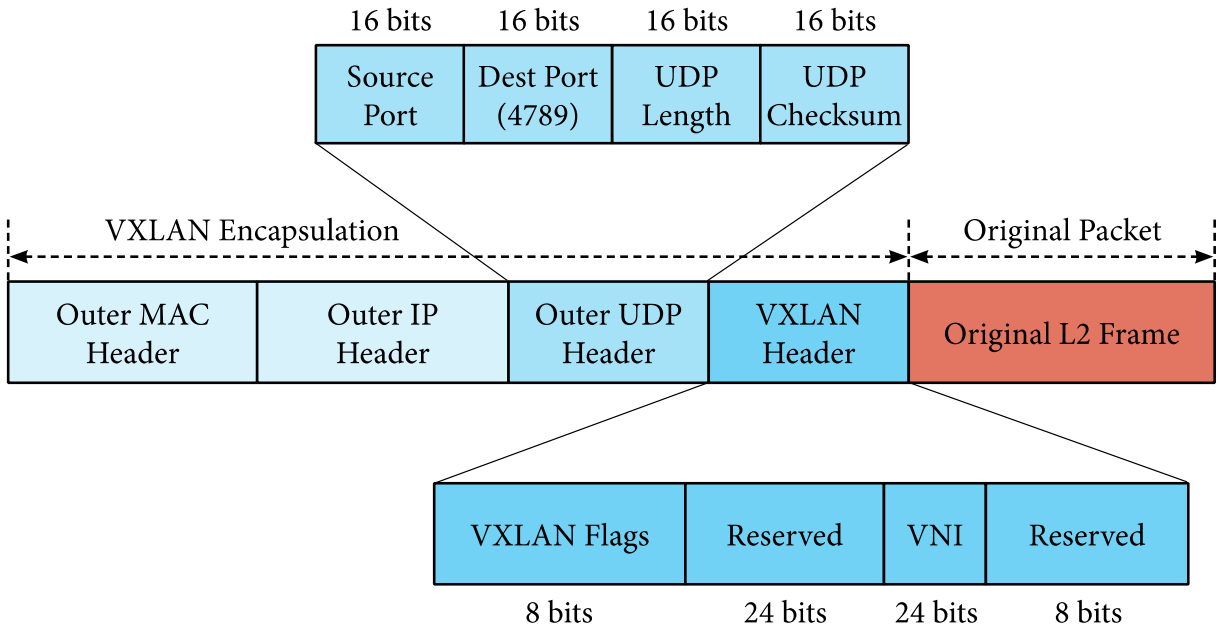
- Virtual eXtensible Local Area Network (RFC-7348)
- designed to address the need for overlay networks within virtualized data centers accommodating multiple tenants
- encapsulates Layer-2 frames into UDP packets

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



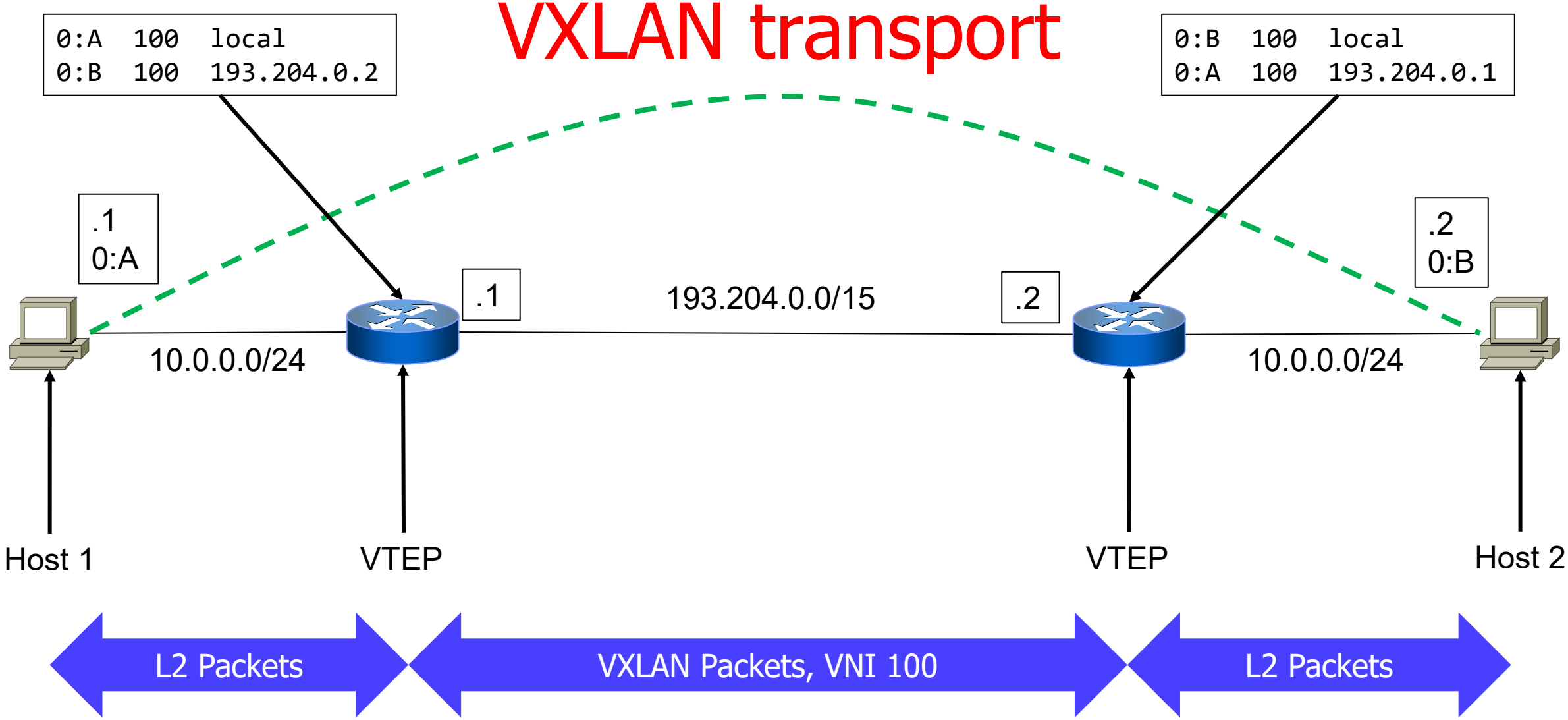
MAC-to-VTEP Table

- 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

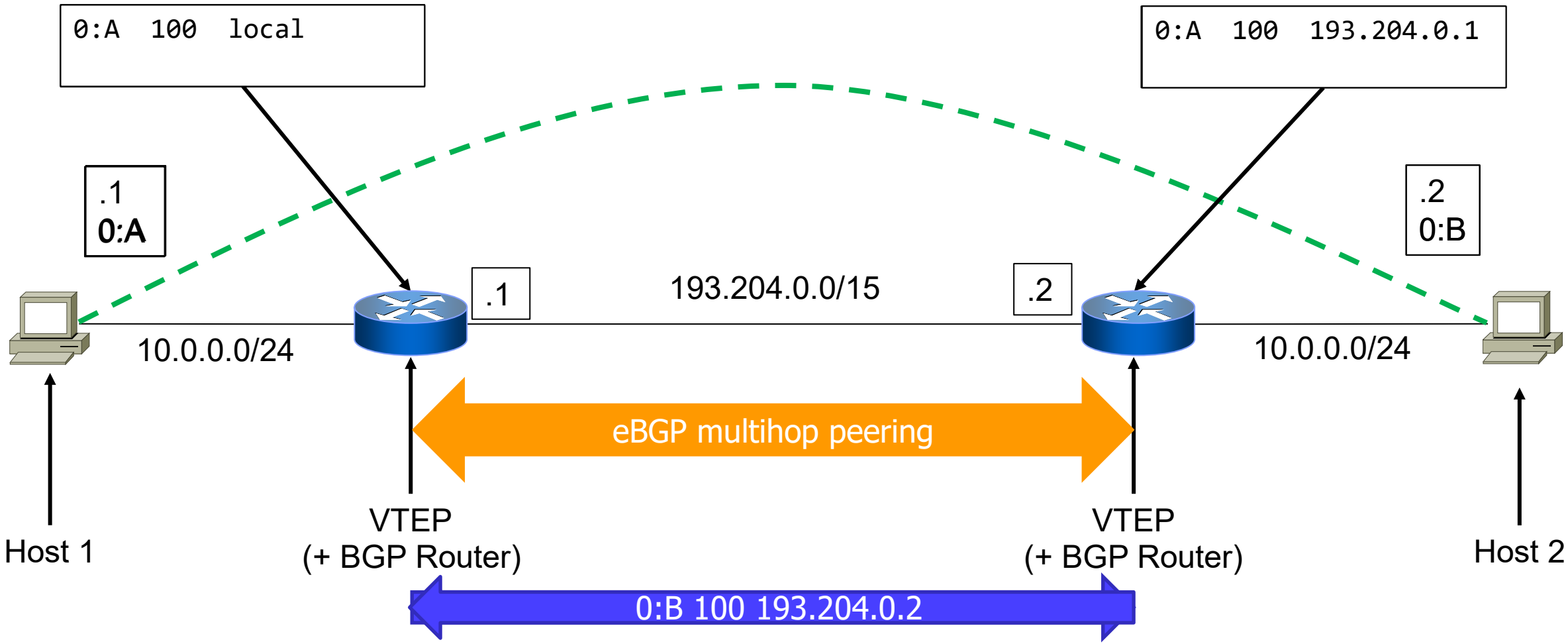
disadvantages of multicast and alternatives

- 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

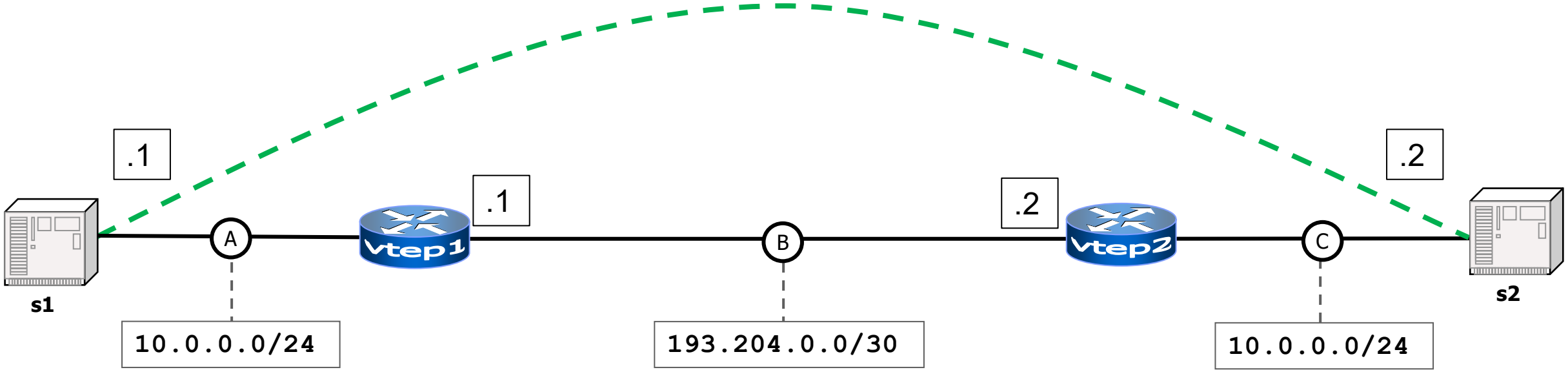
VXLAN and EVPN-BGP transport



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

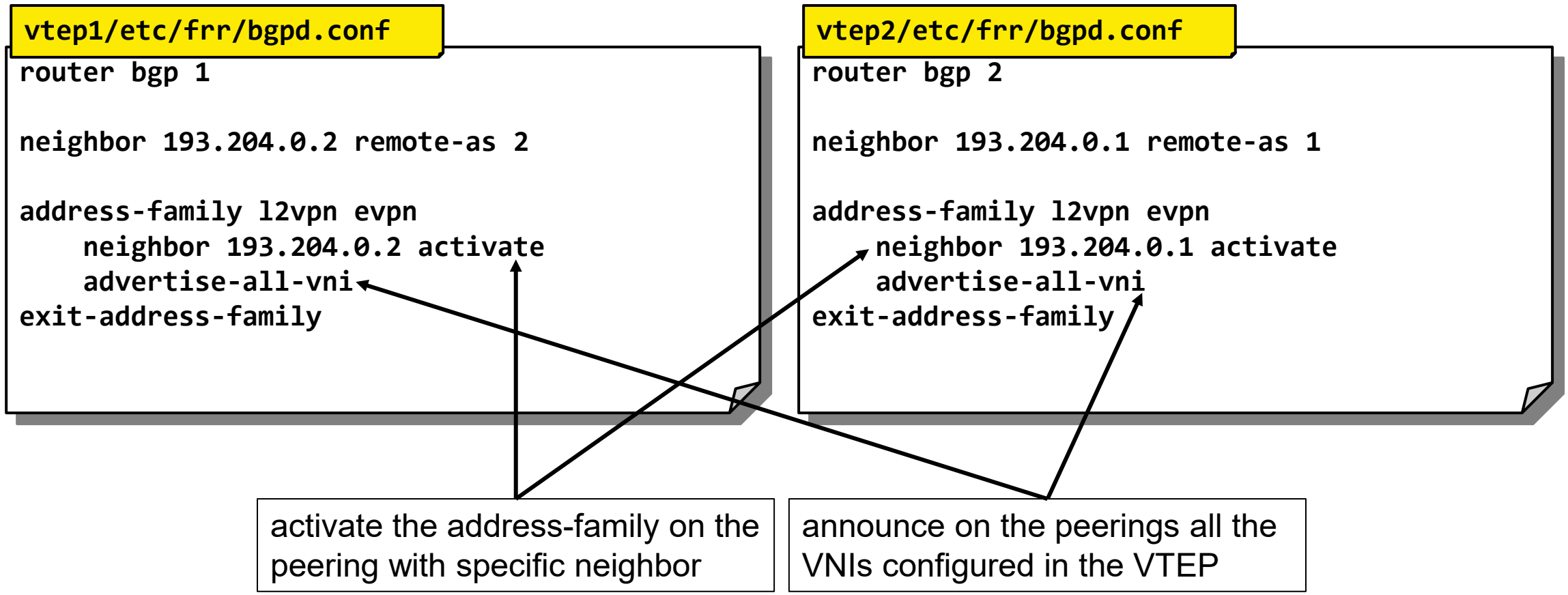
vtep e-BGP configuration

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
```



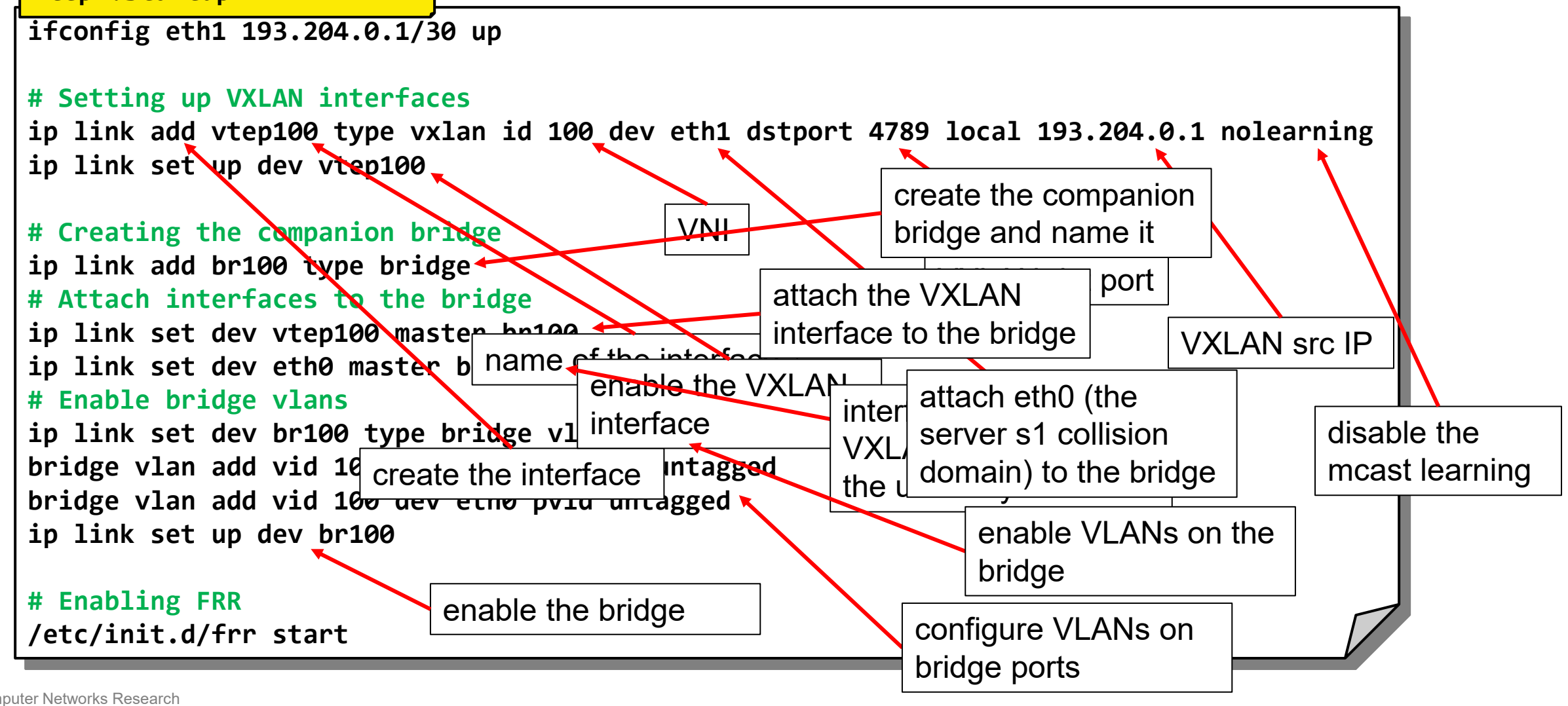
companion bridge

- 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
bridge vlan add vid 100 dev eth0 pvid untagged
ip link set up dev br100

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



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

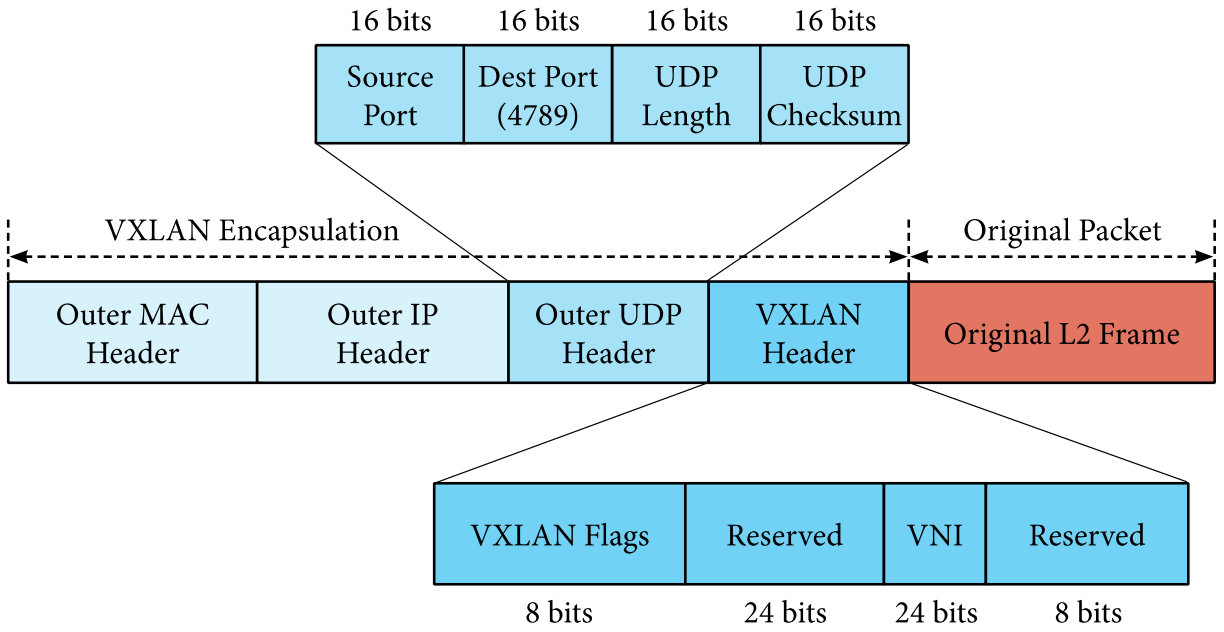
VTEP destination

MAC address of s1

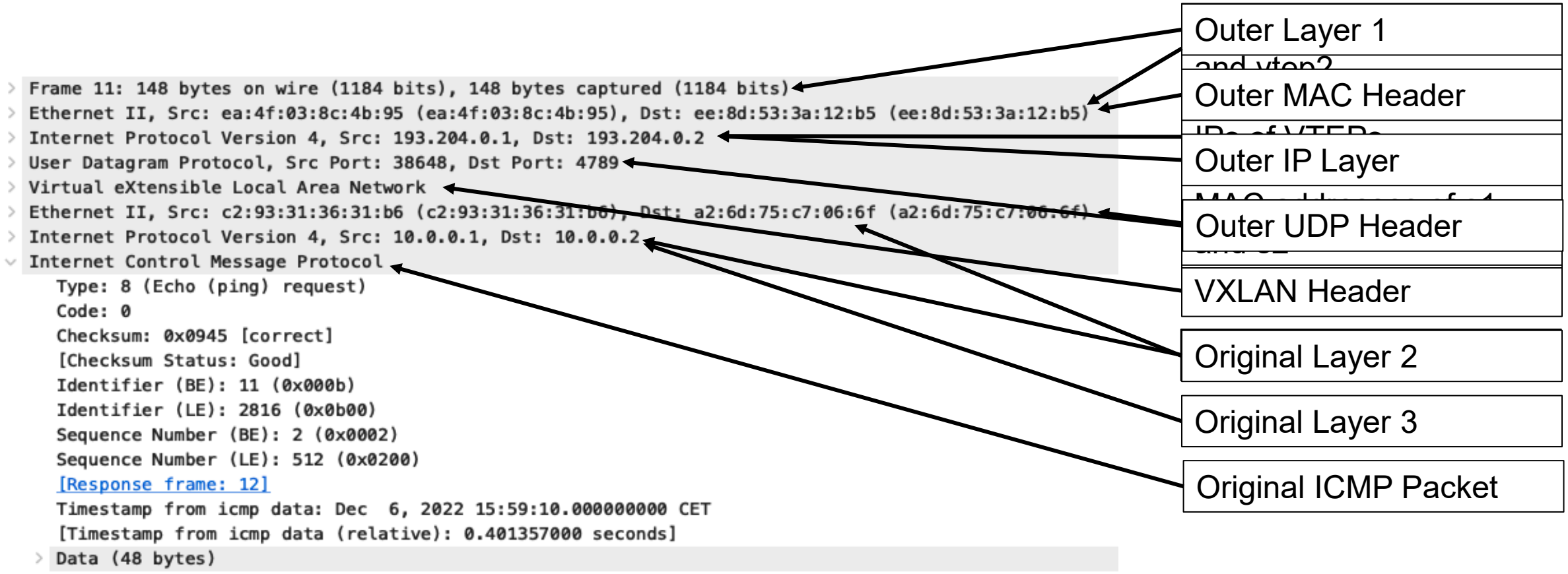
VNI of s1

VXLAN encapsulation

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



a PING packet encapsulated in VXLAN



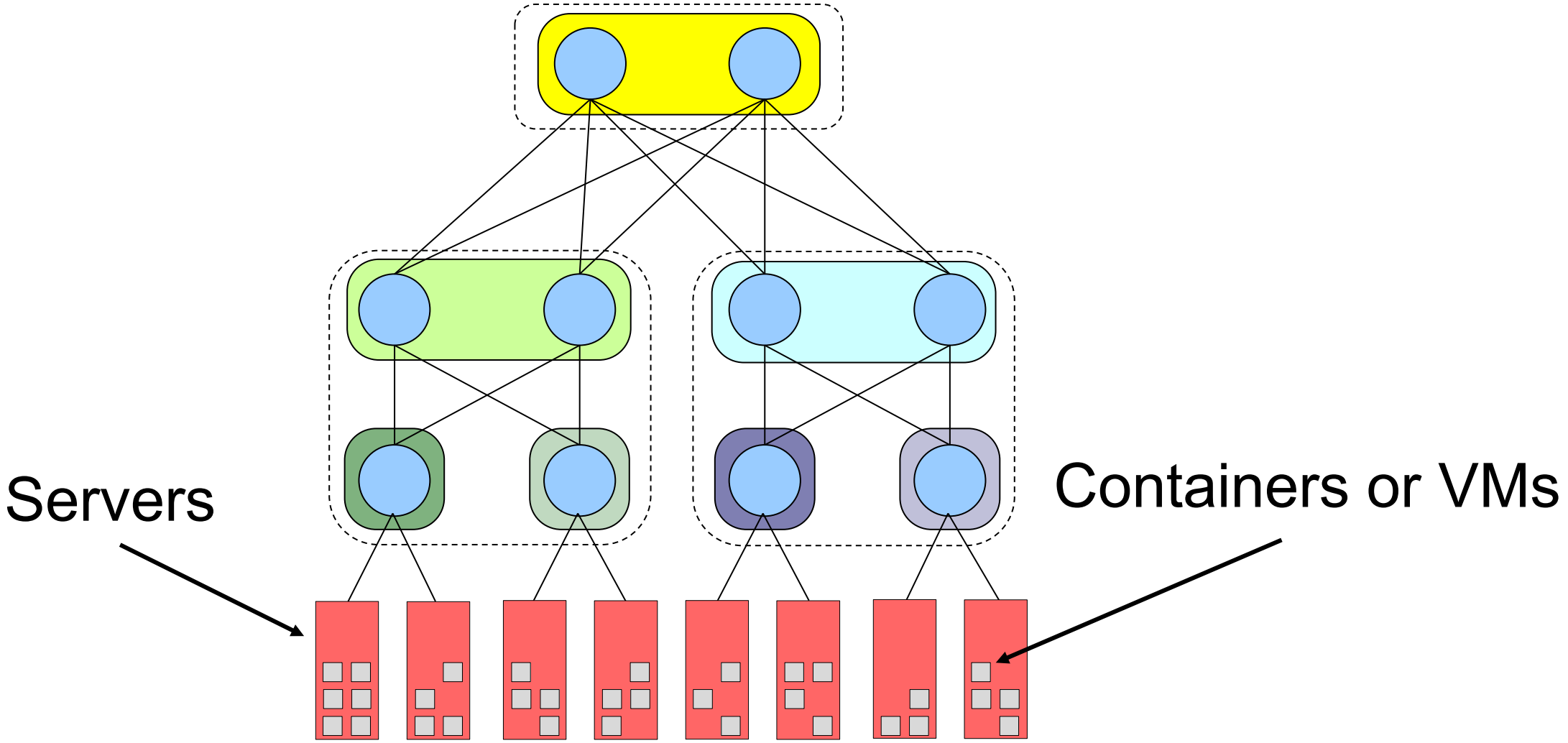
putting together

(EVPN-)BGP 😉

overview

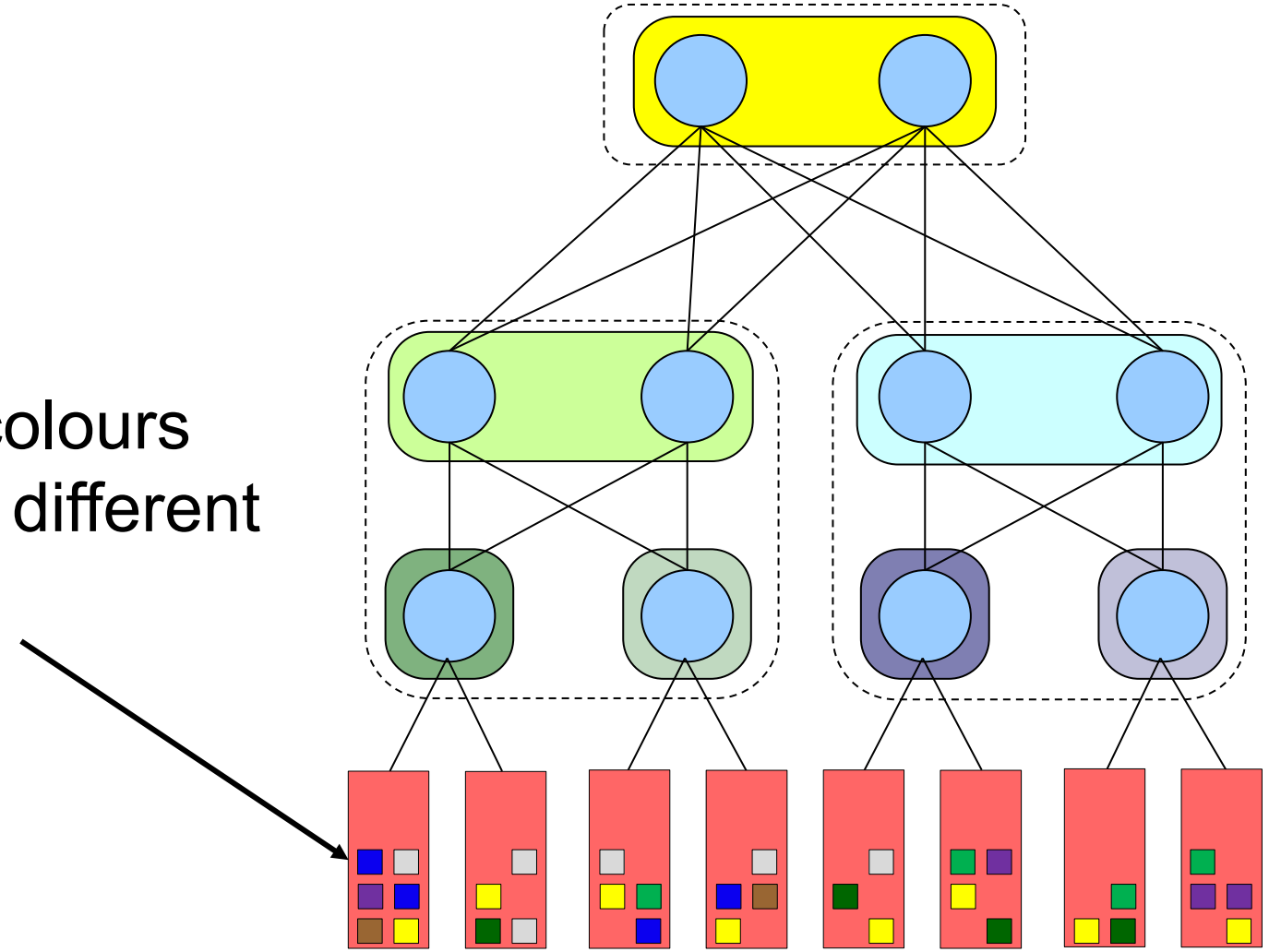
- containers or VMs of different tenants
- where is the VTEP?
- the Leaf-server links
- inside the servers
- dual attached servers
- a complete lab experience

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
  - in each Leaf
    - 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

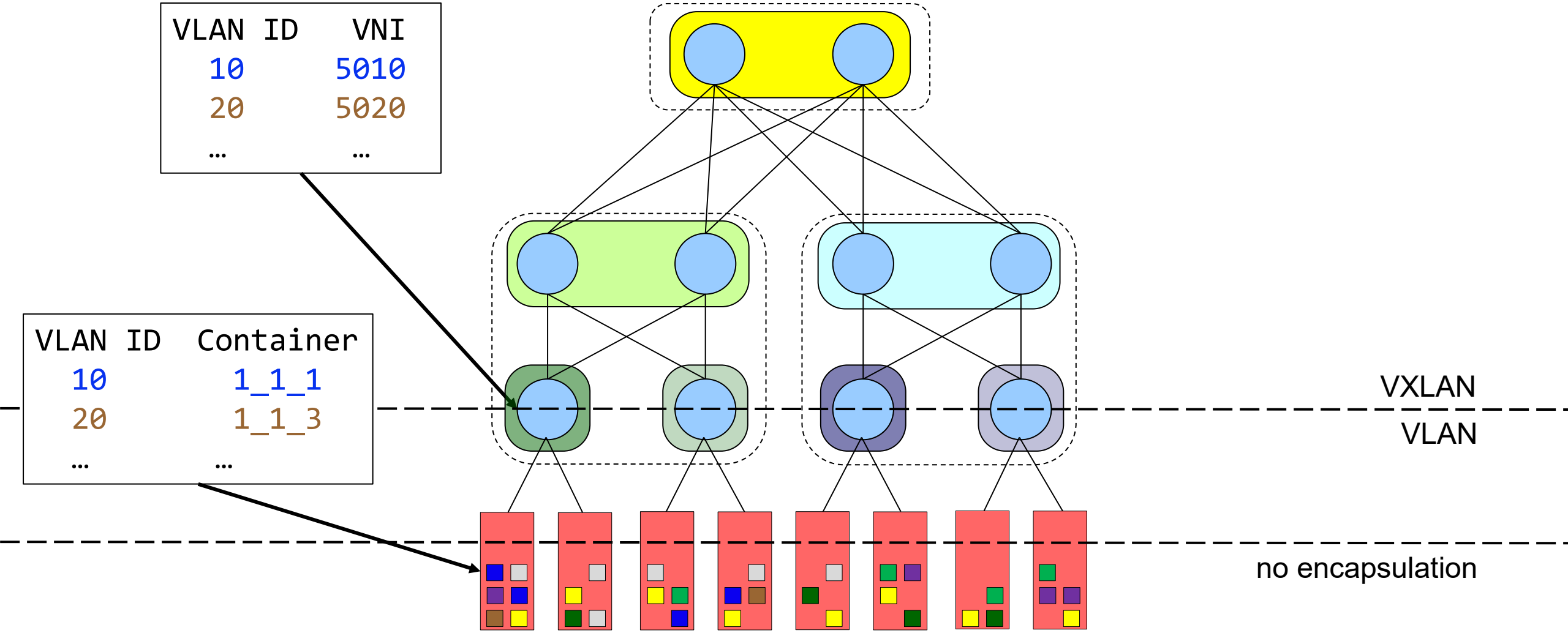
inside the Leaves

- 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



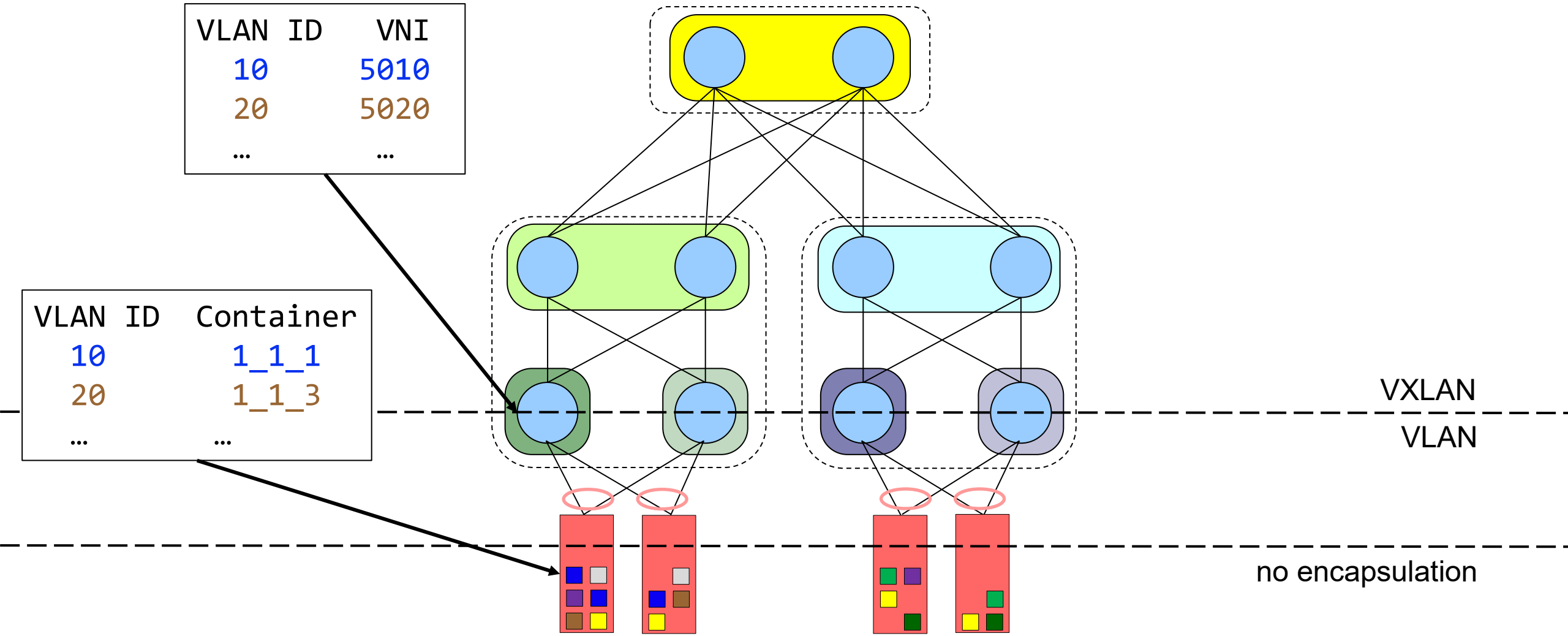
the last problem to overcome

- 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



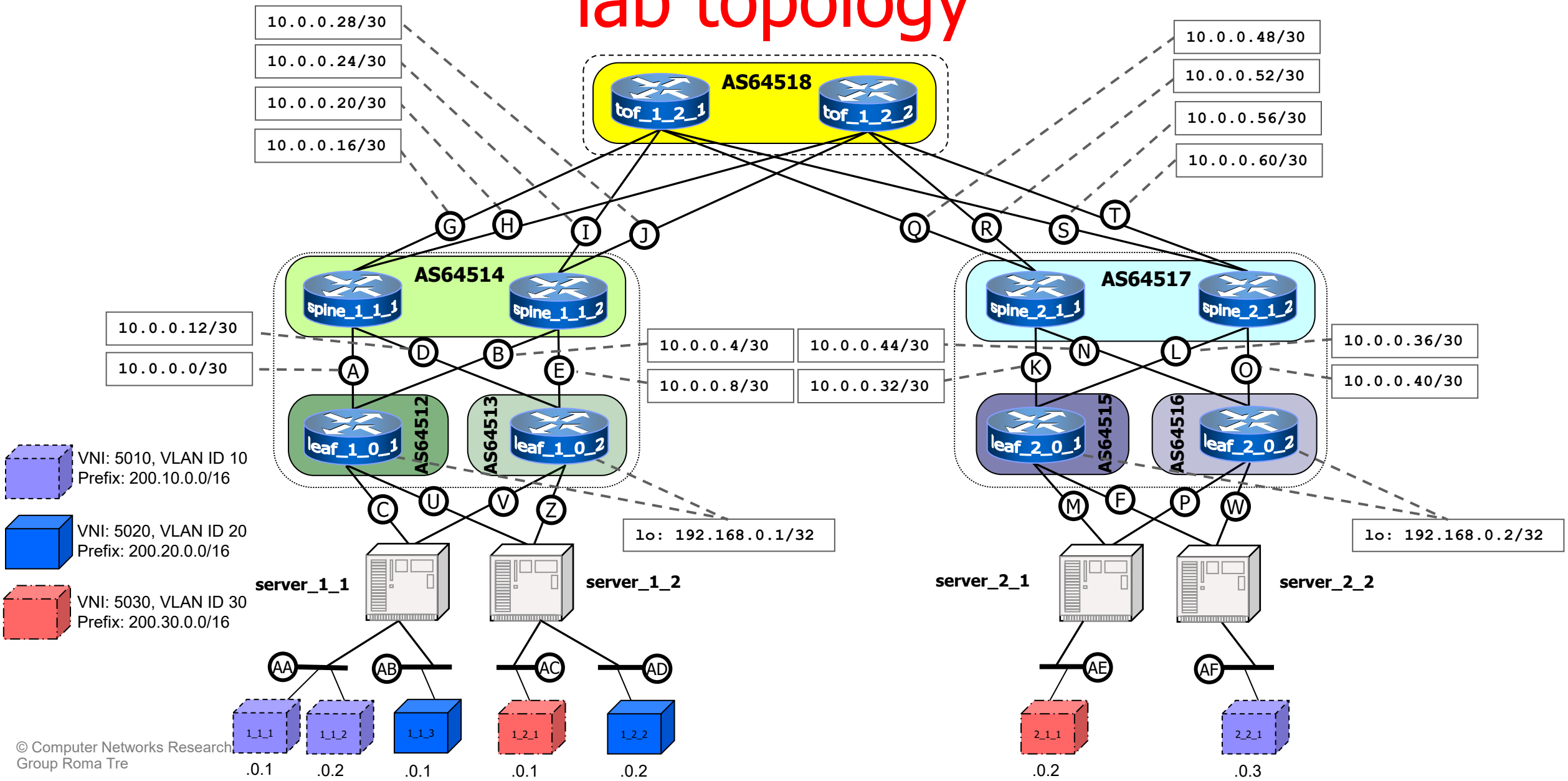
EVPN-BGP Fat-Tree lab

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
```

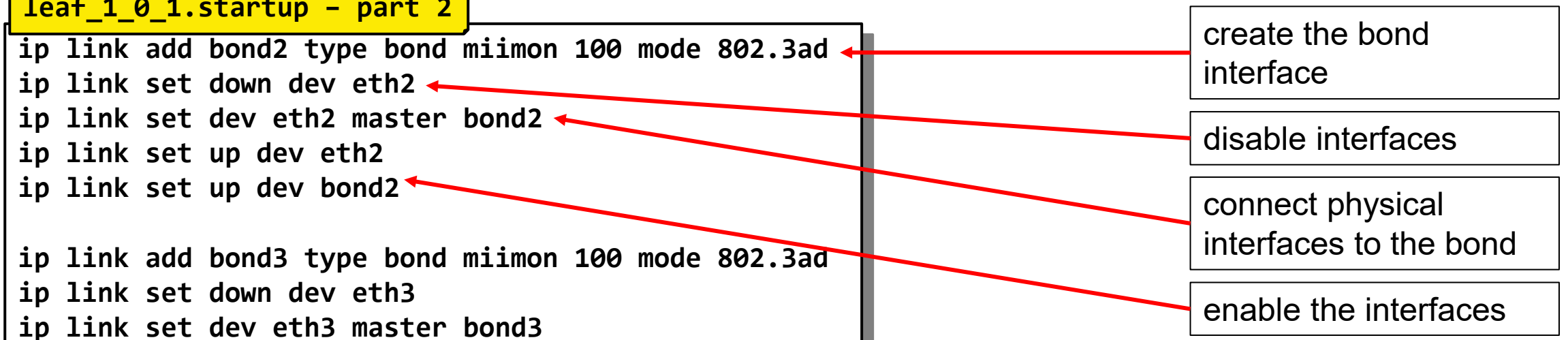
configure an anycast IP address on the loopback interface

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
```



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 eth2 master br100
ip link set dev eth3 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 eth2
bridge vlan add vid 20 dev eth2
bridge vlan add vid 20 dev eth3
bridge vlan add vid 30 dev eth3
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