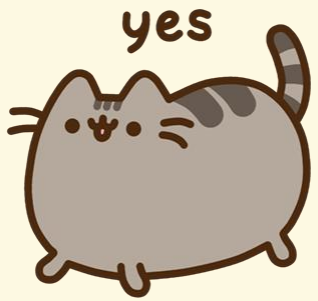
Normando Marcolongo | Micso s.r.l.

VXLAN/EVPN in un piccolo DC

Tutorial | Case study

ITNOG8 | Bologna 20240507

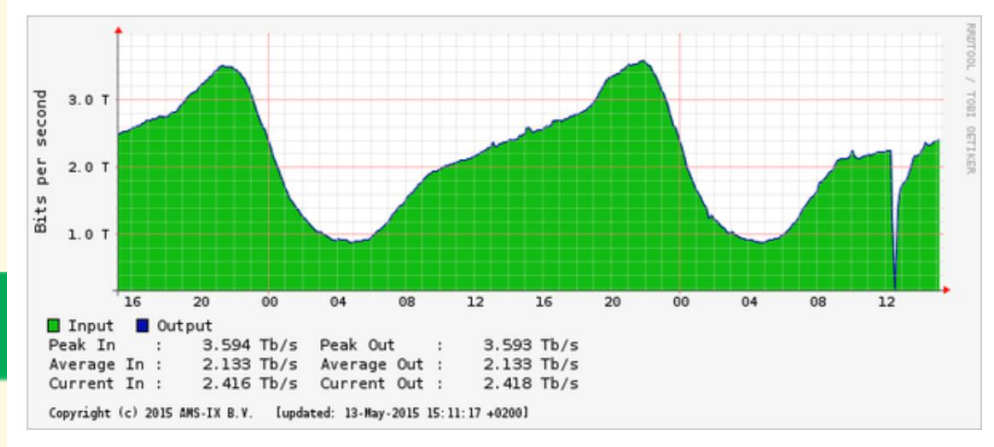
2 VXLAN || ! VXLAN

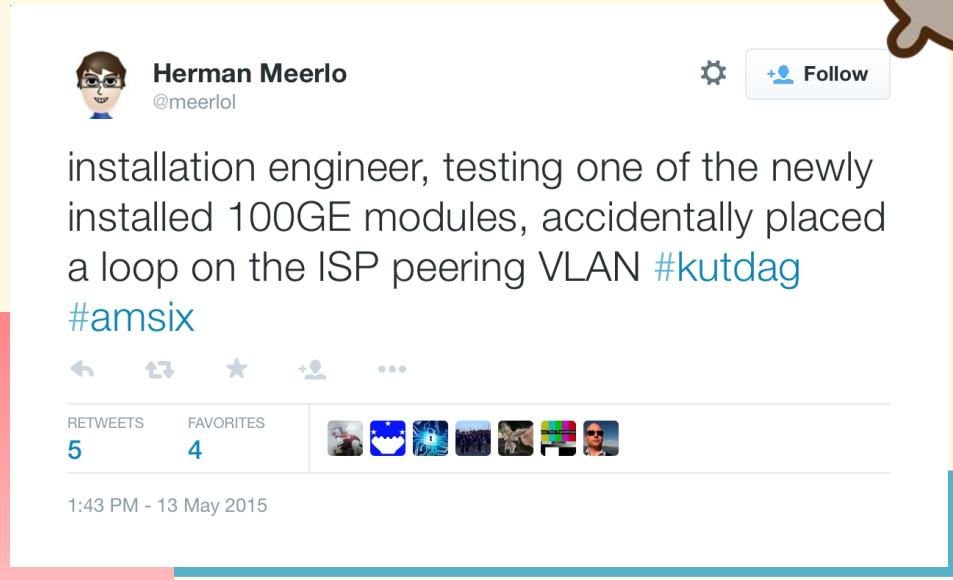


- Ricerca di un miglioramento
- Automatizzare sfruttando nuovi switch
- Noia
- Una domenica c’è stato un loop
- Non voglio saper usare STP

- Costi
- Paura di una nuova tecnologia
- Downtime
- Ho già MLAG (vPC su Cisco)
- So usare STP

Perché STP è brutto !!??





- Blocked alternate paths
- Forward-on-failure behaviour
- Slow link establishment
- Full of kludges
- Kludges bad implemented
- **Not tolerant to human error**

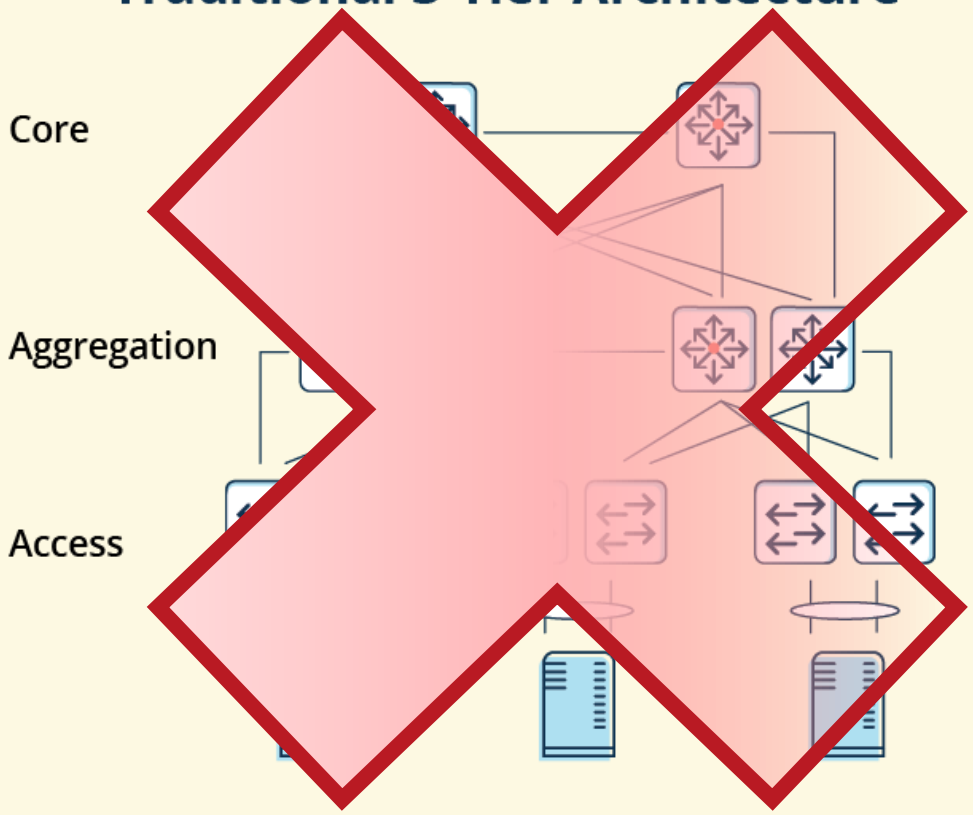
Source: https://blog.ipspace.net/2016/03/spanning-tree-protocol-stp-and-bridging.html

Perché VXLAN è bello

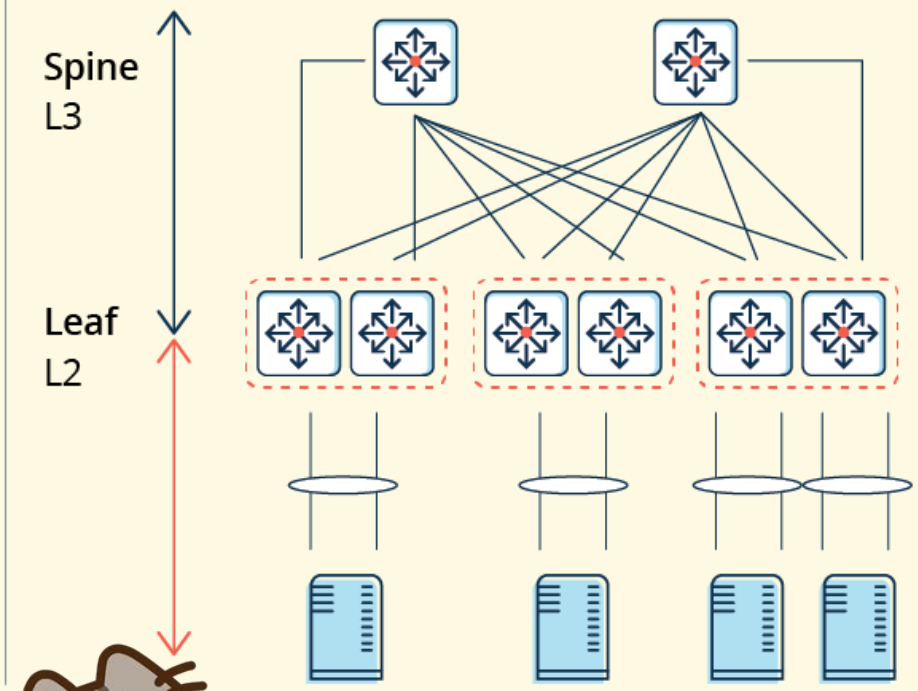
- No dependency on MLAG
- «We know what we’re doing» behaviour
- Do you prefer routing over bridging? We do!
- Based on decades old underlying protocols (OSPF, IS-IS, BGP)
- **Explicit complexity better than hidden complexity**

Source: https://blog.ipspace.net/2020/02/pragmatic-evpn-designs.html e https://blog.ipspace.net/2020/03/should-i-go-with-vxlan-or-mlag-with-stp.html

Traditional 3-Tier Architecture



2-Tier Spine-Leaf Architecture



6 Le parole di VXLAN/EVPN

- La nomenclatura che userò è basata su Cisco
- OSPF tipicamente per noi piccoli (IS-IS, chiaramente per tutti gli altri) usato per «**l'underlay**»: la comunicazione tra gli switch

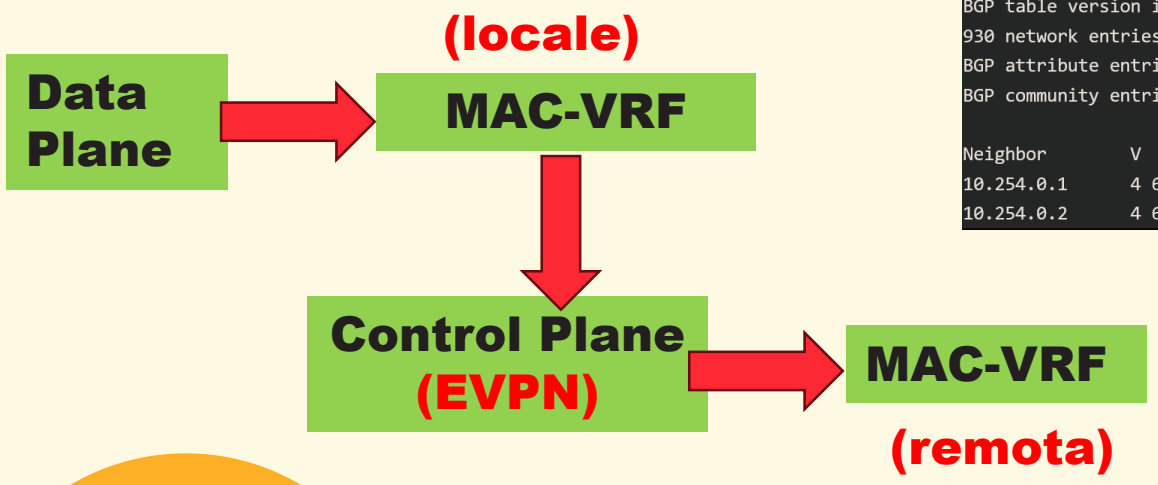
```
leaf1# sh ip ospf neighbors
 OSPF Process ID UNDERLAY VRF default
 Total number of neighbors: 2
 Neighbor ID     Pri State            Up Time  Address         Interface
 spine1            1 FULL/ -          22w2d    10.254.1.1      Eth1/53
 spine2            1 FULL/ -          22w2d    10.254.1.5      Eth1/54

leaf2# sh ip ospf neighbors
 OSPF Process ID UNDERLAY VRF default
 Total number of neighbors: 2
 Neighbor ID     Pri State            Up Time  Address         Interface
 spine1            1 FULL/ -          22w2d    10.254.1.9      Eth1/53
 spine2            1 FULL/ -          22w2d    10.254.1.13     Eth1/54
..
..
```

7 Le parole di VXLAN/EVPN

- Il **data plane** di uno switch *apprende i MAC* e *«popola»* ***localmente*** *una* ***MAC address-table*** **(MAC-VRF)**

8 Le parole di VXLAN/EVPN

- Il **control plane** per mezzo dell'**EVPN**, estensione del BGP per trasportare informazioni sulla raggiungibilità degli endpoint (come gli indirizzi MAC Layer 2), *diffonde questa informazione su tutta l'infrastruttura*




```
leaf1# show bgp l2vpn evpn summary
BGP summary information for VRF default, address family L2VPN EVPN
BGP router identifier 10.254.0.129, local AS number 65001
BGP table version is 475294, L2VPN EVPN config peers 2, capable peers 2
930 network entries and 1439 paths using 242160 bytes of memory
BGP attribute entries [293/50396], BGP AS path entries [0/0]
BGP community entries [0/0], BGP clusterlist entries [6/24]

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.254.0.1      4 65001  273776  210322   475294    0    0    22w2d 382
10.254.0.2      4 6500
```

```
   Network            Next Hop            Metric     LocPrf     Weight Pat
Route Distinguisher: 10.254.0.129:32802    (L2VNI 10035)
*>l[2]:[0]:[0]:[48]:[0010.f35c.f956]:[0]:[0.0.0.0]/216
                                  254.0.129                          100      32768 i
*>l[2]:[0]:[0]:[48]:              10.f35c.faea]:[0]:[0.0.0.0]/216
                                  10.254.0.129                       100      32768 i
```

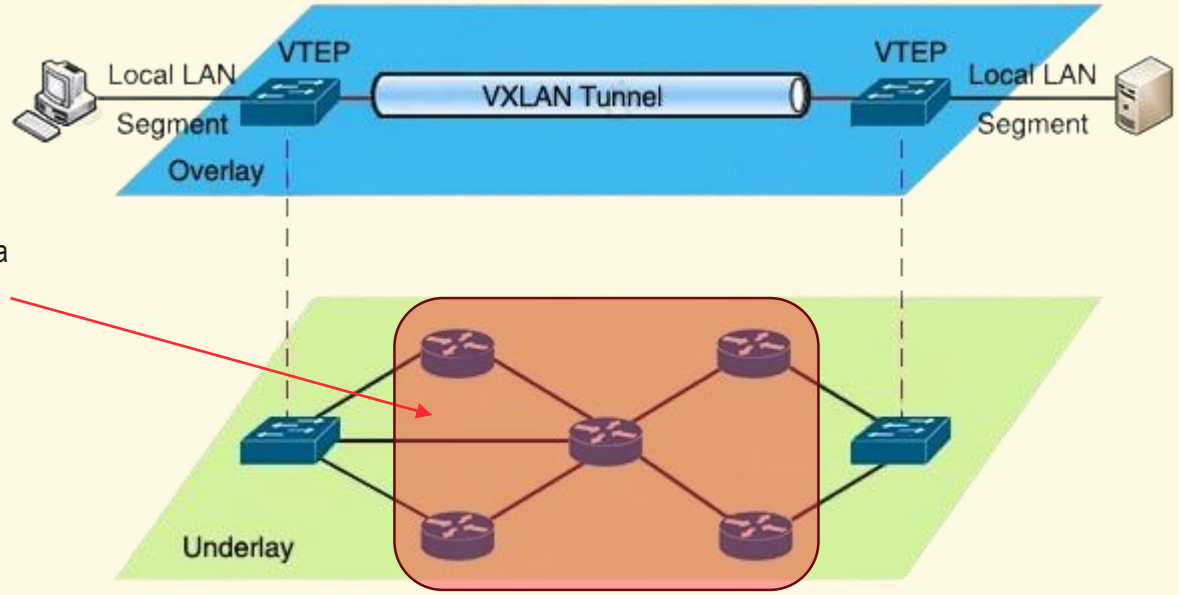
9 Le parole di VXLAN/EVPN

- **VXLAN** (**overlay**) *incapsula* i frame Ethernet Layer 2 in *pacchetti IP* usando a L4 *UDP* (WKP 4789)

```
Frame 7848: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits) on interface ens192, id 1
Ethernet II, Src: 00:ad:e6:88:1b:08, Dst: 00:ad:b3:fd:1b:08
Internet Protocol Version 4, Src: 203.0.113.1, Dst: 203.0.113.4
User Datagram Protocol, Src Port: 62492, Dst Port: 4789
Virtual eXtensible Local Area Network
    Flags: 0x0800, VXLAN Network ID (VNI)
    Group Policy ID: 0
    VXLAN Network Identifier (VNI): 10000
    Reserved: 0
Ethernet II, Src: 00:50:56:ad:85:06, Dst: 00:50:56:ad:7d:68
Internet Protocol Version 4, Src: 198.51.100.11, Dst: 198.51.100.44
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xeabc [correct]
    [Checksum Status: Good]
    Identifier (BE): 60 (0x003c)
    Identifier (LE): 15360 (0x3c00)
    Sequence Number (BE): 1 (0x0001)
    Sequence Number (LE): 256 (0x0100)
    [Response frame: 7849]
    Timestamp from icmp data: Feb 24, 2024 08:12:54.931440000 Romance Standard Time
    [Timestamp from icmp data (relative): 0.002592388 seconds]
    Data (40 bytes)
```

10 Le parole di VXLAN/EVPN

- *Il dispositivo che esegue l'incapsulamento e il decapsulamento* VXLAN è chiamato endpoint del tunnel VXLAN (**VTEP**): tipicamente è «dentro» lo switch *leaf*



Qui immaginate una topologia L&S ma non ho trovato una immagine decente. VXLAN, ovviamente, funziona anche su topologie degeneri di qualunque tipo.

(*) **VTEP**: **V**XLAN **T**unnel **E**nd-**P**oint

11 Le parole di VXLAN/EVPN

- In una rete overlay VXLAN, ogni sottorete o segmento Layer 2 è *identificato* in modo univoco da un identificatore di rete virtuale (**VNI**) ≈ VID

```
Frame 7848: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits) on interface ens192, id 1
Ethernet II, Src: 00:ad:e6:88:1b:08, Dst: 00:ad:b3:fd:1b:08
Internet Protocol Version 4, Src: 203.0.113.1, Dst: 203.0.113.4
User Datagram Protocol, Src Port: 62492, Dst Port: 4789
Virtual eXtensible Local Area Network
    Flags: 0x0800, VXLAN Network ID (VNI)
    VXLAN Network Identifier (VNI): 10000
Ethernet II, Src: 00:50:56:ad:85:06, Dst: 00:50:56:ad:7d:68
Internet Protocol Version 4, Src: 198.51.100.11, Dst: 198.51.100.44
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xeabc [correct]
    [Checksum Status: Good]
    Identifier (BE): 60 (0x003c)
    Identifier (LE): 15360 (0x3c00)
    Sequence Number (BE): 1 (0x0001)
    Sequence Number (LE): 256 (0x0100)
    [Response frame: 7849]
    Timestamp from icmp data: Feb 24, 2024 08:12:54.931440000 Romance Standard Time
    [Timestamp from icmp data (relative): 0.002592388 seconds]
    Data (40 bytes)
```

24 bit!

(*) **VNI**: **V**XLAN **N**etwork **I**dentifier

12 VXLAN/EVPN sia! Cosa serve?

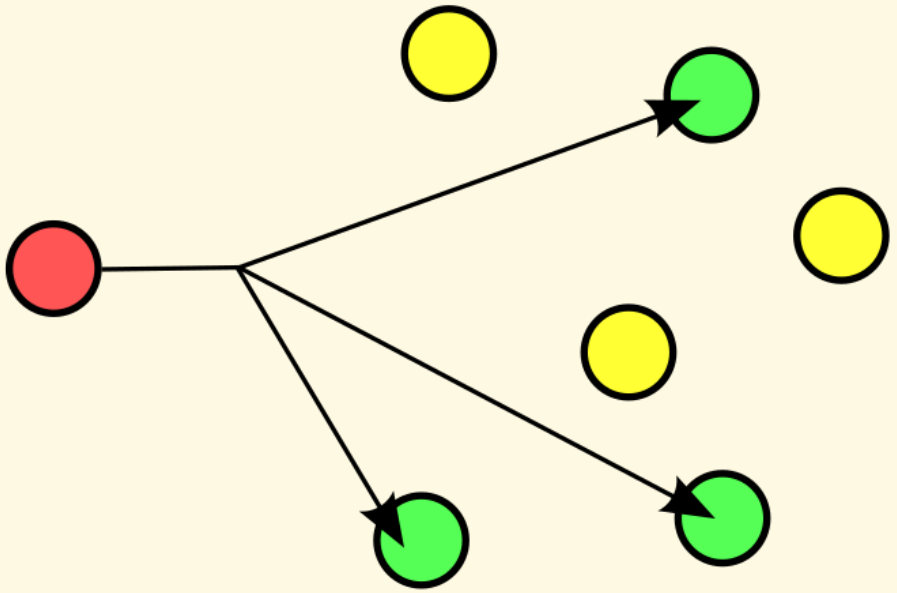
- Abbiamo scelto Cisco Nexus 9000 per la facilità di reperire hardware sul «mercato grigio» a costi assolutamente accessibili

- Ci sono molte cose da tener presente *in ogni caso*:
  - **Spine: switch «stupidi», tanta capacità**
  - **Leaf: switch «smart», eventualmente meno capacità**
  - **VTEP in hardware**
  - Supporto/licenze

13 Traffico B.U.M.

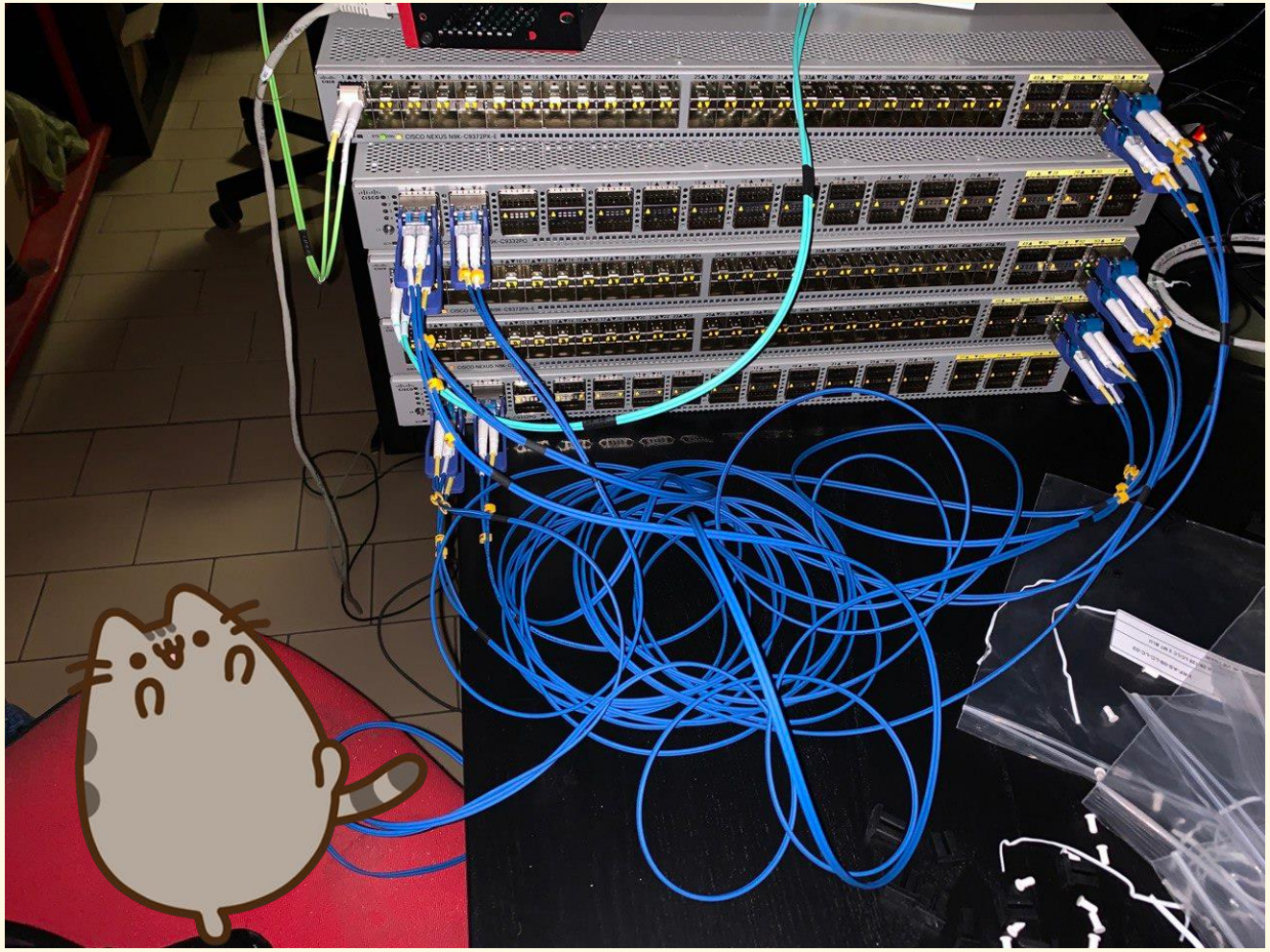
Broadcast, Unknown Unicast, Multicast (BUM)

Un frame di questo tipo **va** inviato ad ogni switch Leaf che ha quello specifico VNI.



- Multicast (PIM-SM, PIM-BiDir)
- **Ingress replication**: Approccio unicast al traffico multi-destinazione, il dispositivo che lo riceve replica ogni pacchetto e lo manda ad ogni dispositivo “egress”

14 Come tutto è iniziato



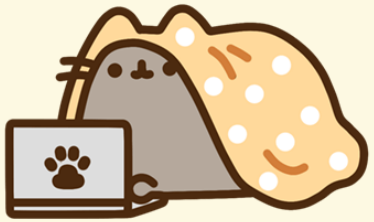
15 Configurazione features

```
! All those features are necessary, the ones in doubt are pointed out.
nv overlay evpn
feature ospf
feature bgp
feature pim
! Maybe not [illegible] my case.
feature interface-v[illegible]
feature lldp
! Could do without :)
feature vtp
```

Solo per RR

Multicast x BUM

```
! All those features are necessary, the ones in doubt are pointed out.
nv overlay evpn
feature ospf
feature bgp
feature pim
! Used for mapping VLANs to VxLAN
feature vn-segment-vlan-based
feature lldp
! Could do without :)
feature vtp
! VTEP
feature nv overlay
```



16 Configurazione routing

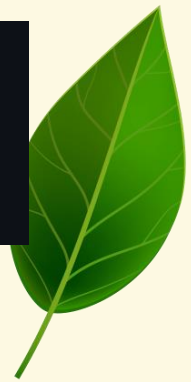
```
interface loopback1
  description RID Loopback
  ip address 10.254.0.1/32
  ip router ospf UNDERLAY area 0.0.0.0
  ip pim sparse-mode

! Pretty clear, isn't it?
router ospf UNDERLAY
  router-id 10.254.0.1
  name-lookup
router bgp 65001
  router-id 10.254.0.1
  address-family l2vpn evpn
  template peer RR-CLIENT
    remote-as 65001
    update-source loopback1
    address-family l2vpn evpn
      send-community extended
      route-reflector-client
  neighbor 10.254.0.64
    inherit peer RR-CLIENT
    description *** Leaf1 ***
  neighbor 10.254.0.65
    inherit peer RR-CLIENT
    description *** Leaf2 ***
```

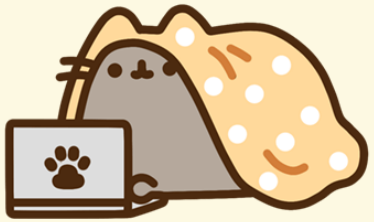
```
interface loopback1
  description RID Loopback
  ip address 10.254.0.64/32
  ip router ospf UNDERLAY area 0.0.0.0

router ospf UNDERLAY
  router-id 10.254.0.64
  name-lookup
router bgp 65001
  address-family l2vpn evpn
  template peer SPINE-RR
    remote-as 65001
    update-source loopback1
    address-family l2vpn evpn
      send-community extended
  neighbor 10.254.0.1
    inherit peer SPINE-RR
    description *** SESSIONE CON SPINE1 (RR) ***
  neighbor 10.254.0.2
    inherit peer SPINE-RR
    description *** SESSIONE CON SP-2 (RR) ***
```

17 Configurazione multicast

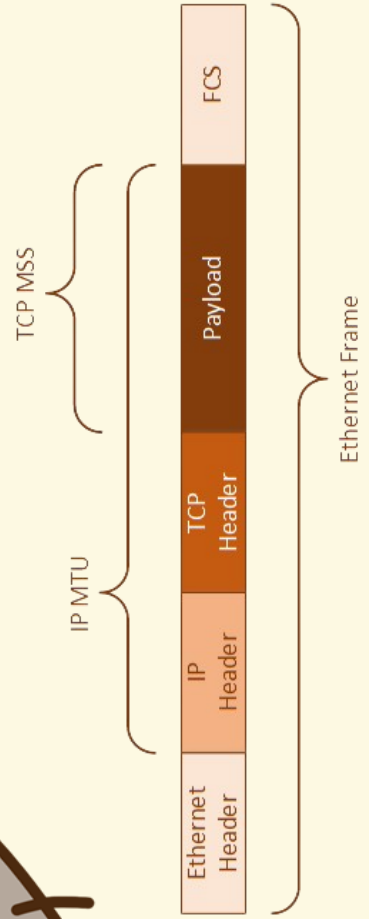
```
ip pim rp-address 10.254.0.1 group-list 224.0.0.0/4
ip pim rp-address 10.254.0.2 group-list 224.0.0.0/4
```

```
! Better use bidir?
! http://blog.reissromoli.com/2019/02/vxlan-flood-learn-lab-test-in-ambiente.html
ip pim rp-address 10.254.0.1 group-list 224.0.0.0/4
ip pim rp-address 10.254.0.2 group-list 224.0.0.0/4
```





18 Configurazione base



NTP, clock, logging, SNMP, management, hardening…
Lasciati come semplice esercizio al lettore

MTU, no però perché è sempre divertente, vero?

```
Outer IPv4 Header ----------- 20 bytes    <==== assumes no extensions are used
Outer UDP Header ------------  8 bytes
VXLAN Header ----------------  8 bytes
Inner Ethernet Frame ------ 1518 bytes (max)
                  |
                   - 14 bytes for header, 4 bytes for 802.1q, 1500 for "inner inner"

                  ------------------------------------
                1554 bytes
```

1600 andrà benone! □

19 Configurazione interfacce

```
interface Ethernet1/1
  description Link Leaf1
  mtu 1600
  ip address 10.254.1.1/30
  ip ospf network point-to-point
  ip router ospf UNDERLAY area 0.0.0.0
  ip pim sparse-mode
  no shutdown

interface Ethernet1/2
  description Link Leaf2
  mtu 1600
  ip address 10.254.1.9/30
  ip ospf network point-to-point
  ip router ospf UNDERLAY area 0.0.0.0
  ip pim sparse-mode
  no shutdown
```

```
interface Ethernet1/53
  description Link Spine1
  no switchport
  mtu 1600
  ip address 10.254.1.2/30
  ip ospf network point-to-point
  ip router ospf UNDERLAY area 0.0.0.0
  ip pim sparse-mode
  no shutdown

interface Ethernet1/54
  description Link Spine2
  no switchport
  mtu 1600
  ip address 10.254.1.6/30
  ip ospf network point-to-point
  ip router ospf UNDERLAY area 0.0.0.0
  ip pim sparse-mode
  no shutdown
```

- Ci sono modi per rendere più carina questa configurazione (es. interfacce `unnumbered`)
- Per ottimizzare LSDB si può usare `ip ospf prefix-suppression`

20 Config. VLAN/VNI & VTEP

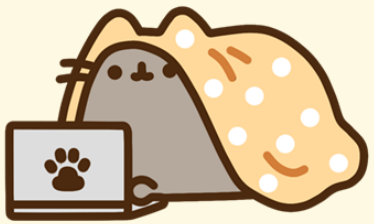
```
vlan 1,82,250,682
vlan 82
  vn-segment 50082
vlan 250
  vn-segment 50250
vlan 682
  vn-segment 50682
```

```
interface loopback2
  description VTEP Loopback
  ip address 10.254.0.129/32
  ip router ospf UNDERLAY area 0.0.0.0
  ip pim sparse-mode
```

```
interface nve1
  no shutdown
  host-reachability protocol bgp
  source-interface loopback2
  ! Multicast groups mapping explained here:
  ! https://community.cisco.com/t5/server-networking/vxlan-vni-to-multicast-g
  member vni 50082
    mcast-group 239.0.0.82
  member vni 50250
    mcast-group 239.0.0.250
  member vni 50682
    ! Ingress replication (https://t.ly/8rAwI)
    ! Nice article here: https://icookservers.blog/2019/08/21/vxlan-mbgp-evpn
    ! And here https://icookservers.blog/2019/09/18/vxlan-mbgp-evpn-with-ingr
    ! ...and here: https://community.cisco.com/t5/data-center-switches/why-vx
    ingress-replication protocol bgp
```

Associo VLAN ↔ VNI

Informo il VTEP della VNI e del relativo gruppo multicast o ingress replication

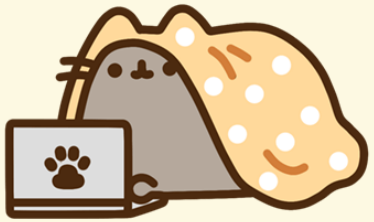


21 Configurazione bridge domain

```
evpn
  vni 50082 l2
    rd auto
    route-target import auto
    route-target export auto
  vni 50250 l2
    rd auto
    route-target import auto
    route-target export auto
  vni 50682 l2
    rd auto
    route-target import auto
    route-target export auto
```

rd: usato per discriminare un possibile overlap di MAC address su VNI differenti. «Agganciato» ad ogni MAC.

route-target: attributo esteso che permette di associare una VNI ad una entry BGP in modo da iniettarla nella **MAC-VRF** giusta e realizzare il bridge

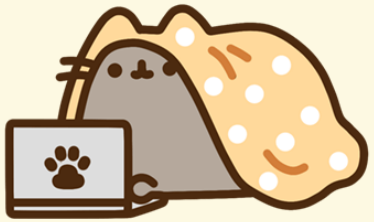


22 La porta di access/trunk

```
interface Ethernet1/1
  switchport mode trunk
  switchport trunk allowed vlan 82,250,682
  no shutdown
```

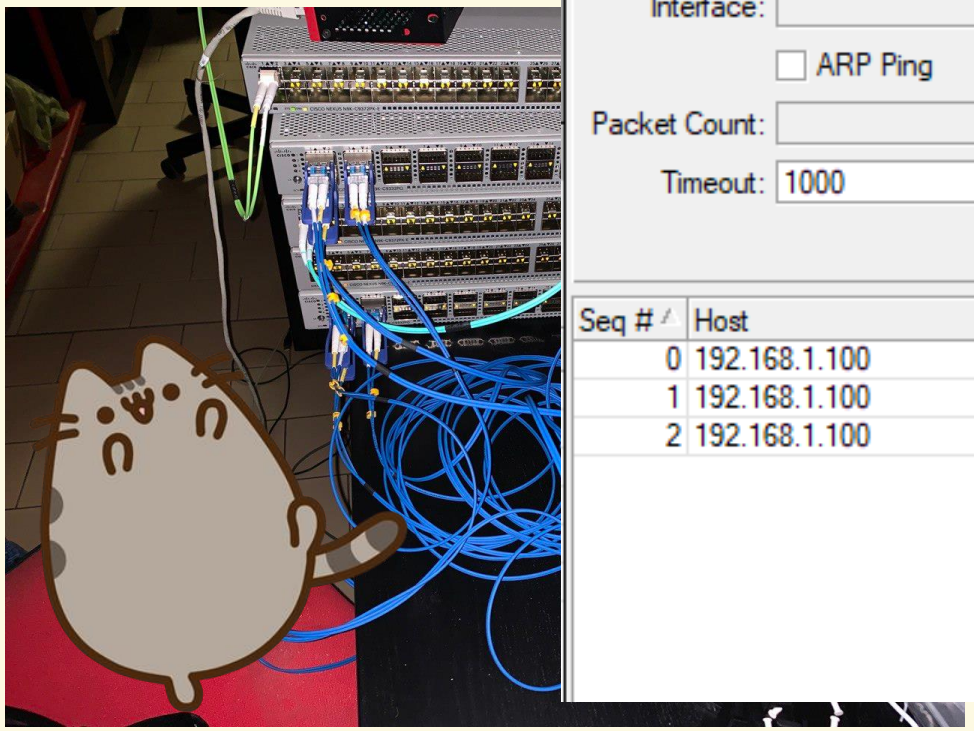
Regardless of what you do in the network core, the VLAN edge ports have the real potential to mess up your network, including the infamous "*let's plug the TX fiber to RX to see if the cable is OK*" layer-1 troubleshooting and "*I wonder if I can solve the bonding on my Windows server by bridging the interfaces together*" approach favored by a CCIE friend of mine.

The only way to protect your network from those stupidities is to use the ancient protection mechanisms available in traditional bridged networks: make the edge switches (VTEPs) STP roots, turn on BPDU guard and root guard, enable storm control…



https://blog.ipspace.net/2019/02/loop-avoidance-in-vxlan-networks.html

23 Come tutto è finito



Ping

General | Advanced

Start
Stop
Close
New Window

Ping To: 192.168.1.100
Interface:
ARP Ping
Packet Count:
Timeout: 1000 ms

Seq #	Host	Time	Reply Size	TTL	Status
0	192.168.1.100	1ms	50	64	
1	192.168.1.100	0ms	50	64	
2	192.168.1.100	0ms	50	64	

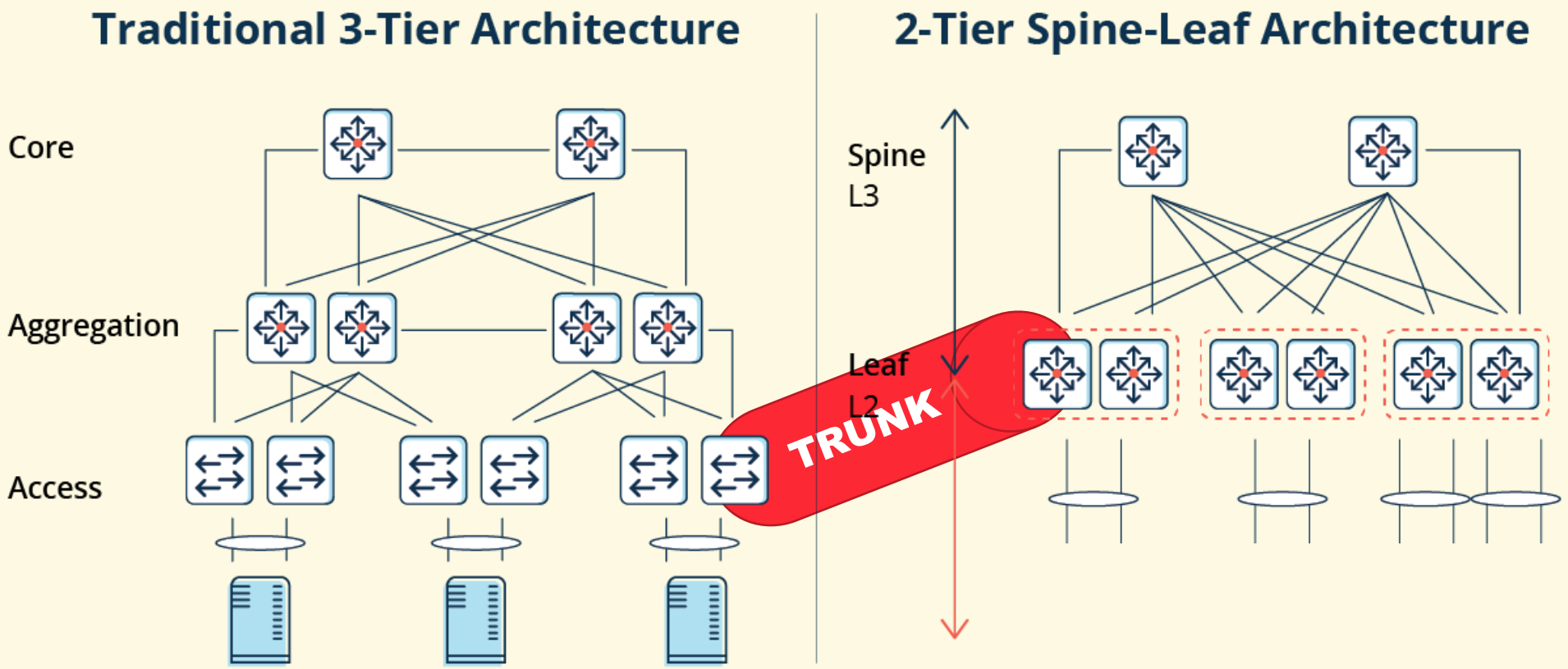
24 Automatizzare

Script python, molto grezzo con autenticazione basata su ‘`db`’ stile `rancid`.

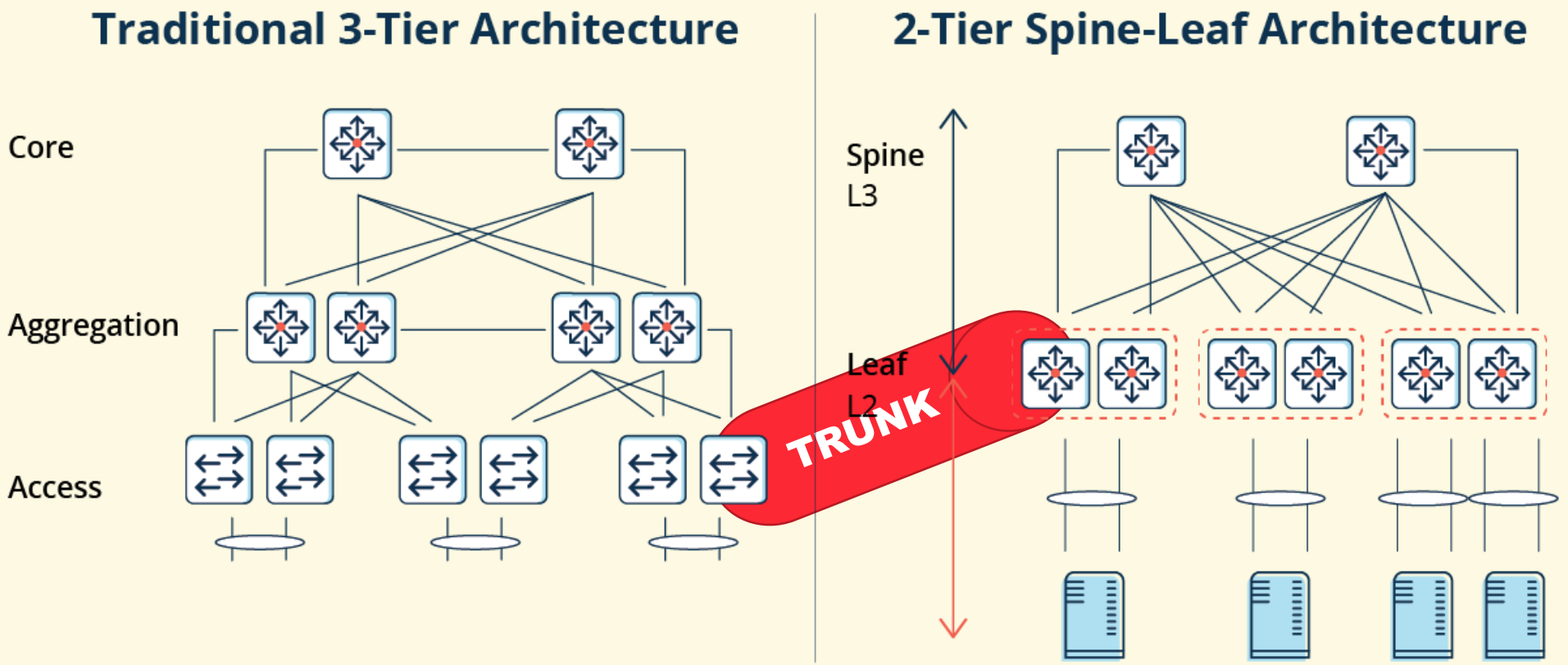
Fork/pull benvenuti!

```
root@rancido:~/nexusvlan# ./createdestroyvxlan.py
2024-04-17 13:38:06 INFO started!
2024-04-17 13:38:06 ERROR create or destroy command needed!
root@rancido:~/nexusvlan# ./createdestroyvxlan.py create
2024-04-17 13:38:09 INFO started!
2024-04-17 13:38:09 ERROR vlan id nedeed!
root@rancido:~/nexusvlan# ./createdestroyvxlan.py create 678
2024-04-17 13:38:12 INFO started!
2024-04-17 13:38:12 ERROR vlan name needed!
root@rancido:~/nexusvlan# ./createdestroyvxlan.py create 678 Test_ITNOG8
2024-04-17 13:38:17 INFO started!
2024-04-17 13:38:17 ERROR switch selection empty!
root@rancido:~/nexusvlan# ./createdestroyvxlan.py create 678 Test_ITNOG8 all
2024-04-17 13:38:20 INFO started!
2024-04-17 13:38:20 INFO VTEP leaf5
2024-04-17 13:38:20 INFO sending commands to switch leaf5
2024-04-17 13:38:20 INFO verifying switch configuration
2024-04-17 13:38:20 INFO VTEP leaf6
2024-04-17 13:38:20 INFO sending commands to switch leaf6
2024-04-17 13:38:20 INFO verifying switch configuration
2024-04-17 13:38:20 INFO ./createdestroyvxlan.py ended!
root@rancido:~/nexusvlan#
```

25 La migrazione



26 La migrazione



27 Horror story: ARP suppression

- ARP suppression is supported for a VNI only if the VTEP host… …st Gateway) for this VNI. The VTEP and SVI for this VLAN must be properly configured for the Distributed Anycast Gateway… …ast gateway MAC address configured and anycast gateway with the virtual IP address on the SVI).

SCAN ME

https://github.com/manolab/vxlanpoc

https://github.com/manolab/nexusvlancreate

- **Configurazioni commentate per spine e leaf con link**
- **Script in Python per automatizzare la creazione/distruzione di VLAN sugli switch**

Grazie!
Domande?

Normando Marcolongo

normando@micso.it