Open Networking for Modern Data Centers Infrastructures: VXLAN Proof-of-Concept Emulation using LNV and EVPN under Cumulus Linux

Gustavo Salazar-Chacón
Facultad de Informática - Posgrados
Universidad Nacional de la Plata – UNLP
La Plata, Argentina
gsalazar.estudios@gmail.com

Luis Marrone
Facultad de Informática - Posgrados
Universidad Nacional de la Plata – UNLP
La Plata, Argentina
lmarrone@linti.unlp.edu.ar

2022 IEEE Sixth Ecuador Technical Chapters Meeting (ETCM) | 978-1-6654-8744-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/ETCM56276.2022.9935681

***Abstract*— Modern telecom service providers need modern Data Centers infrastructures. One Overlay encapsulation protocol brings the ease, flexibility, and good-response behavior to support technological changes oriented to support customer requirements: "VXLAN". This work presents a VXLAN proof-of-concept emulation using the Open Networking paradigm and confirm the mentioned characteristics of the protocol through two mechanisms: The not-so-common process called LNV (Lightweight Network Virtualization) and EVPN (Ethernet VPN) for large scale implementations, bringing scalable and low-cost but high-performance solutions.**

Keywords—VXLAN; LNV; EVPN; BGP; Open Networking

I. Introduction

Businesses are facing the Digital Transformation Era, accelerated due to the Covid-19 pandemic, wars, and economic recession. Moreover, in [1] explains that IT is evolving towards a cloud consumption model, modifying the way applications are being designed and implemented, in fact, according to [2], those challenges are leading to an evolution in the design of Data Centers (DCs) architectures.

According to [1], aspects like flexibility, resiliency, Multitenant capacity, excellent performance, and scalability should be addressed to support the new demands of rapid-growing cloud consumption.

Modern DCs have evolved to Spine-Leaf architecture and started to implement ways to encapsulate traffic and form effective Overlay tunnels, as seen in Fig. 1, thus, making a simpler and more agile infrastructures, as well as using open mechanisms to generate interaction between geographically separated data centers, and that is why Open Networking comes into play [12].

Open Networking is part of the evolutionary process of the Networks, allowing for full-standardization, full-use of technological resources at an adequate economic value (better ROI and TCO), as well as a better and adequate adaptation to changes [11].

Cumulus Linux, as a NOS (Network Operating System), part of the Open Networking ecosystem, has proven to be one of the cornerstones in the world of open technology, for this reason, knowing its use and configuration is imperative for professionals and researchers in the IT field that are developing and implementing modern DC Architectures.

Finally, VXLAN Fabric with LNV and VXLAN Fabric with BGP EVPN control plane, are the solutions proposed in this research to implement VXLAN. The proof-of-Concept scenarios will demonstrate the configuration feasibility of VXLAN in a business-NOS Linux environment and could be used as a guide to implement it in real-world scenarios at a lower-cost comparing with vendor-centric solutions.

II. Problem Statement

Lack of correct IP Addressing to segment different DCs, IPv4 address depletion, poor VLAN segmentation leading towards poor scalability, not to mention the limited number of VLANs available with 12-bits tag field in IEEE 802.1Q Header are the main problems regarding Data Centers connectivity, not just in large Service Providers networks and DCs, but also in any modern telecommunication infrastructure.

With the purpose to give an elegant solution to these problems, a PoC (Proof-of-Concept) is carried out to demonstrate the VXLAN feasibility of implementation in an Open Network environment.

III. Justification and Importance

According to [5], DC infrastructures are built by a combination of routing and switching protocols with overlay encapsulation techniques towards a Software-Defined architecture (Fig. 1). However, implement those techniques could generate multiple points of failure if the correct framework of protocols is not well-selected.

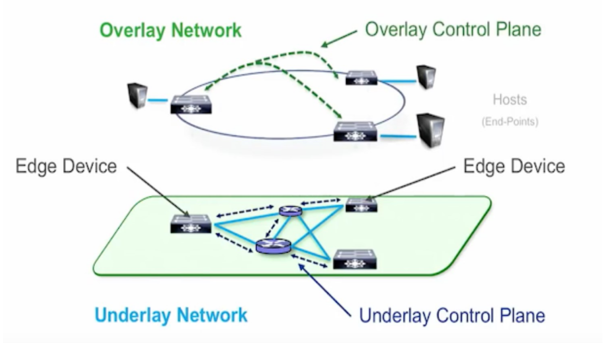


Fig. 1. Underlay-Overlay Paradigm [3]

At the beginning of the programmatic era in networking, many IT vendors started with the idea of OTV in DC environments. Overlay Transport Virtualization (OTV) is a technology that extends L2 capabilities between different Data Centers, or in more simple words, how DC could interconnect which each other: Routing MAC-based data by encapsulating it within IP packets. Fig. 2 shows how OTV works at a High-level scenario.

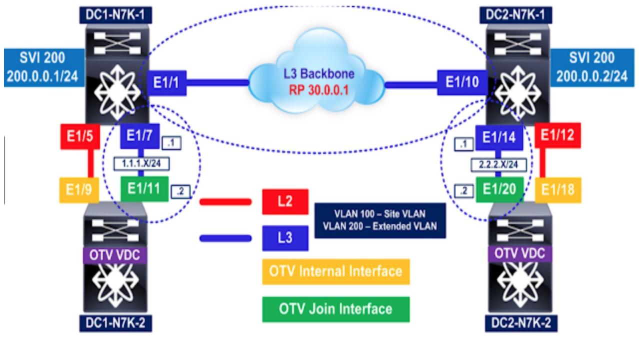


Fig. 2. OTV Schema [6]

Perhaps, it is a good way to send traffic across a transport network or L3-Backbone, however, the complexity of the encapsulation and the hardware needed makes OTV not the best option. VXLAN, as seen in Fig. 3, tries to solve many of the mentioned problems, bringing a new paradigm: A L2 tunnel across a L3 transport network, connecting two VLANs geographically separated (East-West DC Communication) as if they are directly connected using the so-called Spine-Leaf Architecture (Fig. 4).

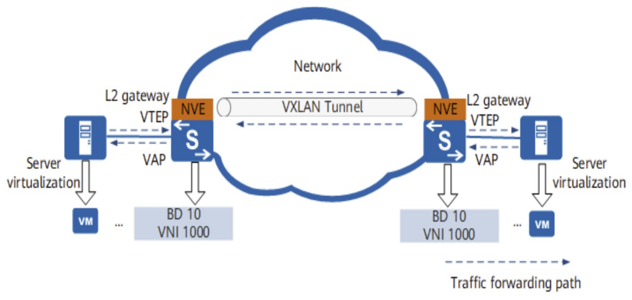


Fig. 3. VXLAN Schema [7]

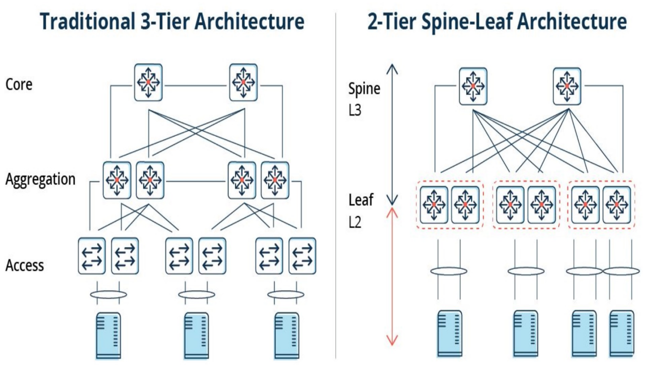

Traditional 3-Tier Architecture
2-Tier Spine-Leaf Architecture
Core
Aggregation
Access
Spine
L3
Leaf
L2

Fig. 4. 2-Tier Spine-Leaf Architecture [4]

IV. Hypothesis

As an advance of the investigation proposed in [1], [5] and [8], this PoC will demonstrate the connection of two physically separated DCs using the same IPv4 subnet and the same VLAN tag on both sites but using two different methods to bring VXLAN up: LNV and EVPN under Open Networking scenario using Cumulus Linux, a low-cost, but high-performance solution.

V. Theoretical Framework

VXLAN, one the most important protocol of the Renaissance Era in Data Networks (see Fig. 5), is changing the way data has been transported between next-generation Data Centers, as well as leading the way in the creation of intelligent L2 over L3 tunnels.

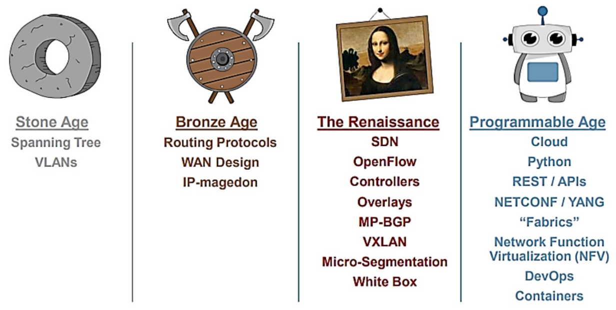


Fig. 5. Networking Eras [5]

A. *VXLAN Fundamentals*

Virtual Extensible LAN or VXLAN, is a relatively new network virtualization technology, which allows solving scalability problems in modern infrastructures and Data Centers.

As mentioned before, Information Technologies are evolving towards a model of consumption of resources in the Cloud, therefore, both, the corporate network, as well as Service Providers and Data Centers, must evolve at the same manner, offering flexibility, resilience, multi-tenant virtualized traffic segmentation capacity, and of course, showing excellent performance.

The reasons for the adoption of VXLAN are:

- Most Data Centers focused on delivering services in the cloud, as well as ISP infrastructures with several clients, face the difficulty of the low scalability of the VLAN Tags (VID), which are up to 4094, which reduces the number of possible segmentations. VXLAN solves this problem by increasing the number of tags to more than 16 million using a 24-bit segment ID.
- The same VLAN cannot span across a Layer 3 domain, but VXLAN can by treating two separate L2 domains and making them appear as one.
- Does not depend on STP (Spanning-tree Protocol) to converge, since it uses Layer 3 protocols.
- Load balancing is performed through all active links, thus avoiding the waste of resources in the infrastructure (Spine – Leaf Architecture).

The first change in the DCs was the migration to a design model called Spine-Leaf, where the Spines are the brain of the network, a scheme very similar to SDN, and the leaves are the access layer to the DC. This architecture makes VXLAN ideal to build SD-Access infrastructure.

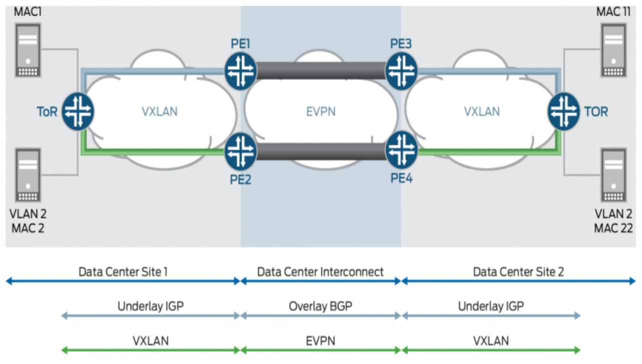


Fig. 6. SD-Access Proposal with VXLAN [9]

B. *VXLAN Encapsulation*

In general terms, VXLAN is a type of Layer 2 (Overlay) VPN that runs on top of a Layer 3 (Underlay) infrastructure.

To achieve this, VXLAN uses an encapsulation similar to traffic with VLANs in that it encapsulates Ethernet MAC addressed frames within IP packets (RFC 7348).

The standard defines an 8-byte (64-bit) header, in which there is a Tunnel identifier called VNID (Virtual Network Identifier) of 24-bits, an identifier used to differentiate the L2 segments and maintain their isolation between them. VXLAN can support up to 224 local segments.

The next figure shows the encapsulation process of an Ethernet Frame within a VXLAN tunnel.

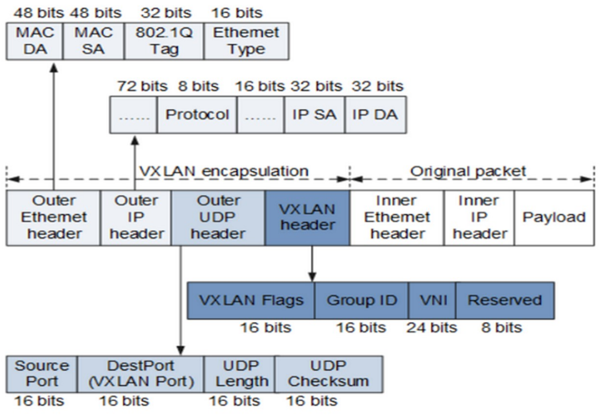


Fig. 7. VXLAN Encapsulation [9]

According to [1], the main VXLAN terminology is:

- **VTEP** (Virtual Tunnel Endpoint): Node that performs the VXLAN tunneling by doing the encapsulation and de-capsulation process. In the 2-tier Spine-Leaf architecture, the VTEP is also known as Leaf.
- **VNID** (Virtual Network Identifier): 24-bit identifier that allows about 16 million logical networks.
- **Bridge-Domain**: Set of physical or logical ports that share the same broadcast domain.

C. *VXLAN PoC: Scenario 1 - LNV*

The configuration of VXLAN through LNV allows joining VTEPs to each other, through a L3 infrastructure.

Static VXLAN tunnels are the simplest solution, being ideal for not excessively large environments. In literature related to this topic, this solution is called LNV (Lightweight Network Virtualization) as it does not include any external controller.

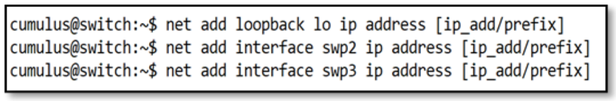
The fundamental process is the mapping between VTEPs and VNIs.

For a basic VXLAN configuration with LNV, it must be considered that:

- VXLAN has a network identifier (VNID). The numbers 0 or 16777215 should not be used as VNIDs, as they are reserved for Cumulus Linux.
- For LNV environments, the VXLAN Registration daemon (VXRD) must be running in all devices.
- LNV can only be used under Cumulus Linux infrastructures.

1) Spine Configuration: The configuration tasks in the Spines are the following:

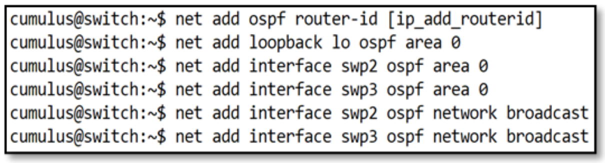
- Configuration of Physical Interfaces (swp#): IP addressing and activation
- Loopback Interface Configuration: It will serve as a device identifier



```
cumulus@switch:~$ net add loopback lo ip address [ip_add/prefix]
cumulus@switch:~$ net add interface swp2 ip address [ip_add/prefix]
cumulus@switch:~$ net add interface swp3 ip address [ip_add/prefix]
```

Fig. 8. VXLAN Configuration: Spines - Addressing

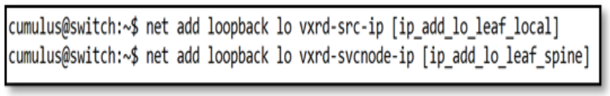
- IGP Configuration as L3 Routing: For the PoC, OSPF is used: Router-id, physical interfaces, and Loopback configuration is made. The OSPF network type is recommended to be broadcast.
- For OSPF to work properly, FRR and the OSPFd daemon must be up and running.
- In the research, it is single-area OSPF.



```
cumulus@switch:~$ net add ospf router-id [ip_add_routerid]
cumulus@switch:~$ net add loopback lo ospf area 0
cumulus@switch:~$ net add interface swp2 ospf area 0
cumulus@switch:~$ net add interface swp3 ospf area 0
cumulus@switch:~$ net add interface swp2 ospf network broadcast
cumulus@switch:~$ net add interface swp3 ospf network broadcast
```

Fig. 9. VXLAN Configuration: Spines – Underlay Routing

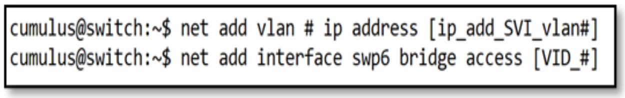
2) Leaf Configuration: The configuration tasks in the Leaves are the same as Spines (Addressing and Underlay Routing), but, on Leaves, it is needed to configure them as Service Nodes in the VXRD process and point to the Spines' IP address.



```
cumulus@switch:~$ net add loopback lo vxrd-src-ip [ip_add_lo_leaf_local]
cumulus@switch:~$ net add loopback lo vxrd-svcnode-ip [ip_add_lo_leaf_spine]
```

Fig. 10. VXLAN Configuration: Leaves – Service Node activation

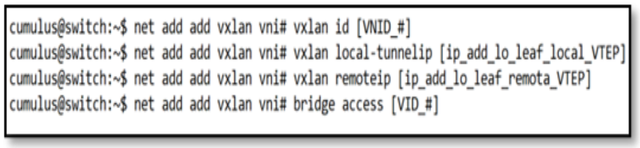
3) VLAN-VXLAN Mapping: VLANs on the Leaf (VTEP) device, must be created and assigned to a specific SWP port. If deemed necessary, SVIs (Switch Virtual Interfaces) can be configured for connectivity testing.



```
cumulus@switch:~$ net add vlan # ip address [ip_add_SVI_vlan#]
cumulus@switch:~$ net add interface swp6 bridge access [VID_#]
```

Fig. 11. VXLAN Configuration: Leaves – VLANs

Finally, the relationship between VNIDs, VIDs and the Loopback's IP addresses of the VTEPs are established with the following commands:



```
cumulus@switch:~$ net add add vxlan vni# vxlan id [VNID_#]
cumulus@switch:~$ net add add vxlan vni# vxlan local-tunnelip [ip_add_lo_leaf_local_VTEP]
cumulus@switch:~$ net add add vxlan vni# vxlan remoteip [ip_add_lo_leaf_remota_VTEP]
cumulus@switch:~$ net add add vxlan vni# bridge access [VID_#]
```

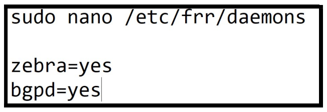
Fig. 12. VXLAN Configuration: Leaves – VNID-VID Mapping

D. *VXLAN PoC: Scenario 2 - EVPN*

At the beginning, RFC 7348, which defines VXLAN, did not include any control plane protocol, and it is just relied on a MAC address learning LNV process.

RFC 7432 defines EVPN (Ethernet Virtual Private Network) as the standard-based control plane for VXLAN.

On Cumulus Linux, the routing environment is installed as part of FRR (Free-Range Routing), so it must be activated (also is needed for underlay routing in LNV scenario).



```
sudo nano /etc/frr/daemons

zebra=yes
bgpd=yes
```

Fig. 13. FRR Activation – Zebra process and BGP

Cumulus Linux fully supports EVPN as the control plane for VXLAN, including for both intra-subnet bridging and inter-subnet routing.

VI. VXLAN Emulation

For LNV and EVPN PoCs, the researchers used GNS3-VM as the emulation software.

A. *LNV Emulation*

For this emulation the topology shown in Fig 14 was used.

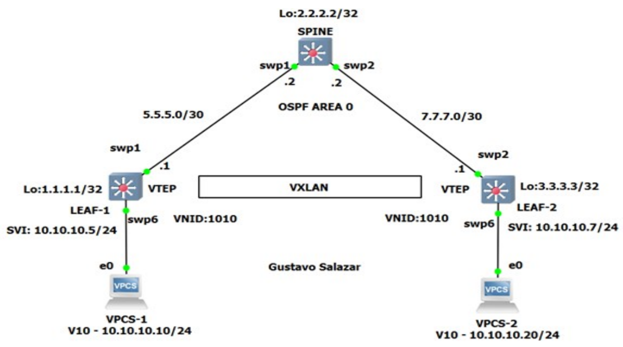
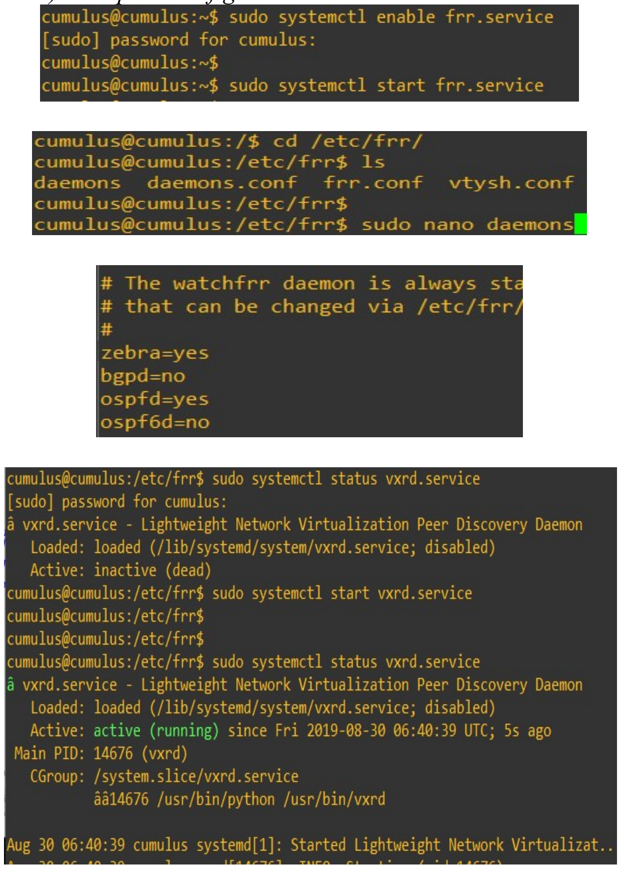


Fig. 14. LNV VXLAN Topology

1) Spine Configuration



```
cumulus@cumulus:~$ sudo systemctl enable frr.service
[sudo] password for cumulus:
cumulus@cumulus:~$
cumulus@cumulus:~$ sudo systemctl start frr.service
```

```
cumulus@cumulus:/$ cd /etc/frr/
cumulus@cumulus:/etc/frr$ ls
daemons  daemons.conf  frr.conf  vtysh.conf
cumulus@cumulus:/etc/frr$
cumulus@cumulus:/etc/frr$ sudo nano daemons
```

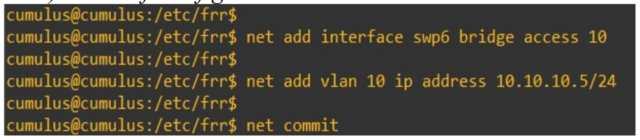
```
# The watchfrr daemon is always sta
# that can be changed via /etc/frr/
#
zebra=yes
bgpd=no
ospfd=yes
ospf6d=no
```

```
cumulus@cumulus:/etc/frr$ sudo systemctl status vxrd.service
[sudo] password for cumulus:
â vxrd.service - Lightweight Network Virtualization Peer Discovery Daemon
   Loaded: loaded (/lib/systemd/system/vxrd.service; disabled)
   Active: inactive (dead)
cumulus@cumulus:/etc/frr$ sudo systemctl start vxrd.service
cumulus@cumulus:/etc/frr$
cumulus@cumulus:/etc/frr$
cumulus@cumulus:/etc/frr$ sudo systemctl status vxrd.service
â vxrd.service - Lightweight Network Virtualization Peer Discovery Daemon
   Loaded: loaded (/lib/systemd/system/vxrd.service; disabled)
   Active: active (running) since Fri 2019-08-30 06:40:39 UTC; 5s ago
 Main PID: 14676 (vxrd)
   CGroup: /system.slice/vxrd.service
           ââ14676 /usr/bin/python /usr/bin/vxrd

Aug 30 06:40:39 cumulus systemd[1]: Started Lightweight Network Virtualizat..
```

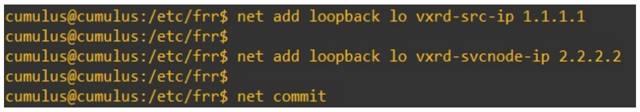
Fig. 15. FRR and VXRD Activation

2) Leaf Configuration



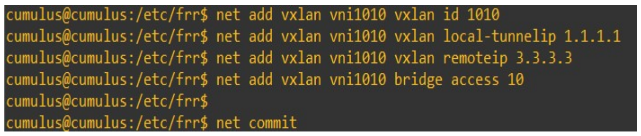
```
cumulus@cumulus:/etc/frr$
cumulus@cumulus:/etc/frr$ net add interface swp6 bridge access 10
cumulus@cumulus:/etc/frr$
cumulus@cumulus:/etc/frr$ net add vlan 10 ip address 10.10.10.5/24
cumulus@cumulus:/etc/frr$
cumulus@cumulus:/etc/frr$ net commit
```

Fig. 16. VLAN Configuration



```
cumulus@cumulus:/etc/frr$ net add loopback lo vxrd-src-ip 1.1.1.1
cumulus@cumulus:/etc/frr$
cumulus@cumulus:/etc/frr$ net add loopback lo vxrd-svcnode-ip 2.2.2.2
cumulus@cumulus:/etc/frr$
cumulus@cumulus:/etc/frr$ net commit
```

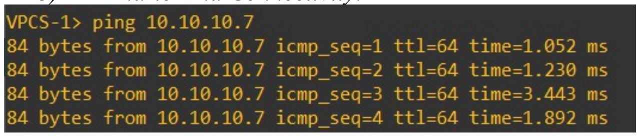
Fig. 17. Service Node Configuration



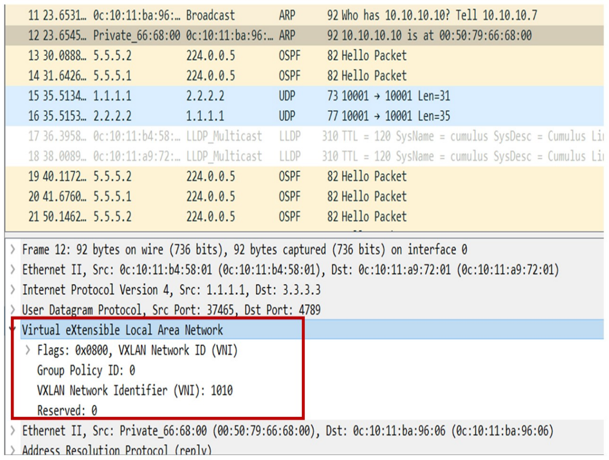
```
cumulus@cumulus:/etc/frr$ net add vxlan vni1010 vxlan id 1010
cumulus@cumulus:/etc/frr$ net add vxlan vni1010 vxlan local-tunnelip 1.1.1.1
cumulus@cumulus:/etc/frr$ net add vxlan vni1010 vxlan remoteip 3.3.3.3
cumulus@cumulus:/etc/frr$ net add vxlan vni1010 bridge access 10
cumulus@cumulus:/etc/frr$
cumulus@cumulus:/etc/frr$ net commit
```

Fig. 18. VNID-VID Mapping

3) End-to-End Connectivity.



```
VPCS-1> ping 10.10.10.7
84 bytes from 10.10.10.7 icmp_seq=1 ttl=64 time=1.052 ms
84 bytes from 10.10.10.7 icmp_seq=2 ttl=64 time=1.230 ms
84 bytes from 10.10.10.7 icmp_seq=3 ttl=64 time=3.443 ms
84 bytes from 10.10.10.7 icmp_seq=4 ttl=64 time=1.892 ms
```



```
11 23.6531… 0c:10:11:ba:96:… Broadcast        ARP   92 Who has 10.10.10.10? Tell 10.10.10.7
12 23.6545… Private_66:68:00 0c:10:11:ba:96:… ARP   92 10.10.10.10 is at 00:50:79:66:68:00
13 30.0888… 5.5.5.2          224.0.0.5        OSPF  82 Hello Packet
14 31.6426… 5.5.5.1          224.0.0.5        OSPF  82 Hello Packet
15 35.5134… 1.1.1.1          2.2.2.2          UDP   73 10001 → 10001 Len=31
16 35.5153… 2.2.2.2          1.1.1.1          UDP   77 10001 → 10001 Len=35
17 36.3958… 0c:10:11:b4:58:… LLDP_Multicast   LLDP 310 TTL = 120 SysName = cumulus SysDesc = Cumulus Li
18 38.0089… 0c:10:11:a9:72:… LLDP_Multicast   LLDP 310 TTL = 120 SysName = cumulus SysDesc = Cumulus Li
19 40.1172… 5.5.5.2          224.0.0.5        OSPF  82 Hello Packet
20 41.6760… 5.5.5.1          224.0.0.5        OSPF  82 Hello Packet
21 50.1462… 5.5.5.2          224.0.0.5        OSPF  82 Hello Packet

> Frame 12: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface 0
> Ethernet II, Src: 0c:10:11:b4:58:01 (0c:10:11:b4:58:01), Dst: 0c:10:11:a9:72:01 (0c:10:11:a9:72:01)
> Internet Protocol Version 4, Src: 1.1.1.1, Dst: 3.3.3.3
> User Datagram Protocol, Src Port: 37465, Dst Port: 4789
v Virtual eXtensible Local Area Network
  > Flags: 0x0800, VXLAN Network ID (VNI)
    Group Policy ID: 0
    VXLAN Network Identifier (VNI): 1010
    Reserved: 0
> Ethernet II, Src: Private_66:68:00 (00:50:79:66:68:00), Dst: 0c:10:11:ba:96:06 (0c:10:11:ba:96:06)
> Address Resolution Protocol (reply)
```

Fig. 19. Ping (without Gateway) and Wireshark Capture

B. *EVPN Emulation*

For this emulation, the topology shown in Fig 20 was used (Based on [1] and [10])

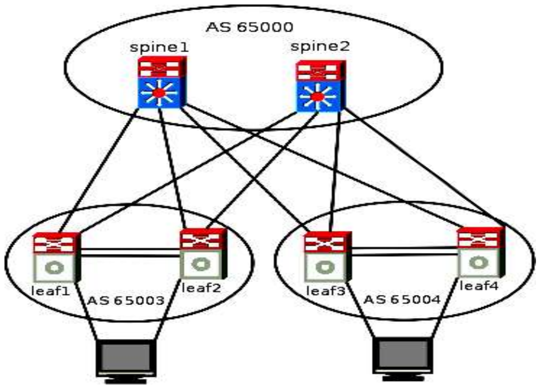


Fig. 20. EVPN VXLAN Topology

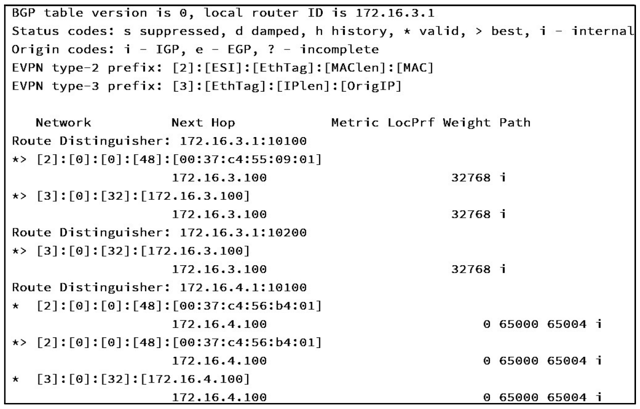
Below are the results obtained for eBGP communication (Fig. 21).

The EVPN routes and a the VXLAN traffic capture with Wireshark are shown in Fig. 22 and Fig. 23 respectively.

```
cumulus@spine1:~$ net show route bgp

show ip route
=============
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, P - PIM, T - Table, v - VNC,
       V - VPN,
       > - selected route, * - FIB route
B>* 172.16.3.1/32 [20/0] via 192.0.2.9, swp1, 08:58:27
B>* 172.16.3.2/32 [20/0] via 192.0.2.11, swp2, 08:57:25
B>* 172.16.3.100/32 [20/0] via 192.0.2.9, swp1, 00:28:29
  *                        via 192.0.2.11, swp2, 00:28:29
B>* 172.16.4.1/32 [20/0] via 192.0.2.13, swp3, 08:56:25
B>* 172.16.4.2/32 [20/0] via 192.0.2.15, swp4, 08:56:10
B>* 172.16.4.100/32 [20/0] via 192.0.2.15, swp4, 00:27:45
  *                        via 192.0.2.13, swp3, 00:27:45
B>* 192.0.2.136/31 [20/0] via 192.0.2.9, swp1, 08:58:27
B>* 192.0.2.138/31 [20/0] via 192.0.2.11, swp2, 08:57:25
B>* 192.0.2.140/31 [20/0] via 192.0.2.13, swp3, 08:56:25
B>* 192.0.2.142/31 [20/0] via 192.0.2.15, swp4, 08:56:10
```

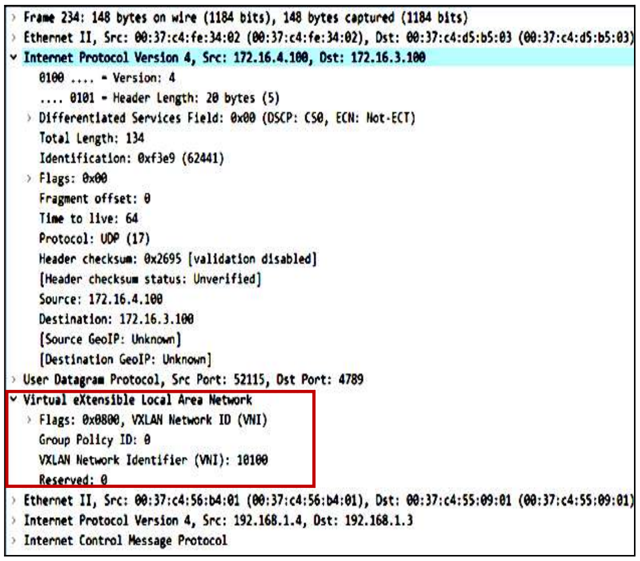
Fig. 21. eBGP Table



```
BGP table version is 0, local router ID is 172.16.3.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
EVPN type-2 prefix: [2]:[ESI]:[EthTag]:[MAClen]:[MAC]
EVPN type-3 prefix: [3]:[EthTag]:[IPlen]:[OrigIP]

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 172.16.3.1:10100
*> [2]:[0]:[0]:[48]:[00:37:c4:55:09:01]
                    172.16.3.100                       32768 i
*> [3]:[0]:[32]:[172.16.3.100]
                    172.16.3.100                       32768 i
Route Distinguisher: 172.16.3.1:10200
*> [3]:[0]:[32]:[172.16.3.100]
                    172.16.3.100                       32768 i
Route Distinguisher: 172.16.4.1:10100
*  [2]:[0]:[0]:[48]:[00:37:c4:56:b4:01]
                    172.16.4.100                           0 65000 65004 i
*> [2]:[0]:[0]:[48]:[00:37:c4:56:b4:01]
                    172.16.4.100                           0 65000 65004 i
*  [3]:[0]:[32]:[172.16.4.100]
                    172.16.4.100                           0 65000 65004 i
```

Fig. 22. EVPN Routes



```
> Frame 234: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits)
> Ethernet II, Src: 00:37:c4:fe:34:02 (00:37:c4:fe:34:02), Dst: 00:37:c4:d5:b5:03 (00:37:c4:d5:b5:03)
v Internet Protocol Version 4, Src: 172.16.4.100, Dst: 172.16.3.100
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 134
    Identification: 0xf3e9 (62441)
  > Flags: 0x00
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
    Header checksum: 0x2695 [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.16.4.100
    Destination: 172.16.3.100
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
> User Datagram Protocol, Src Port: 52115, Dst Port: 4789
v Virtual eXtensible Local Area Network
  > Flags: 0x0800, VXLAN Network ID (VNI)
    Group Policy ID: 0
    VXLAN Network Identifier (VNI): 10100
    Reserved: 0
> Ethernet II, Src: 00:37:c4:56:b4:01 (00:37:c4:56:b4:01), Dst: 00:37:c4:55:09:01 (00:37:c4:55:09:01)
> Internet Protocol Version 4, Src: 192.168.1.4, Dst: 192.168.1.3
> Internet Control Message Protocol
```

Fig. 23. VXLAN Analysis with Wireshark

VII. Results and Contributions

After the process and execution of the Proof-of-Concepts, the feasibility of using VXLAN in scenarios than can be expanded to modern DC environments was verified.

Using Open-Hardware (like Mellanox, Broadcom, among others) and Open NOS like Cumulus Linux will reduce the TCO (Total Cost of Ownership), will lower the cost of OPEX (Operational Expenditures), and will obtain an adequate ROI (Return of Investment), because those technologies can take steps towards full network automation (NetDevOps), customer connectivity (Leaf) can go from 1Gbps to 100Gbps, thus producing the "Next-Generation Switching over IP Fabrics" with total visibility.

According to [13], "Cumulus Linux enables you to do more with current resources at 1/3 the initial price of the competition. And rapid service delivery means that where it used to take 24 hours, customers can now get their own network ready within 10 minutes."

VIII. Conclusions

This research presents the main reasons why is necessary to implement VXLAN in modern DCs, but without neglecting the theoretical part of the protocol and the technologies used to develop the PoCs: LNV and EVPN.

The two PoCs demonstrate the VXLAN feasibility of configuration under Open Networking scenarios, connecting two geographically disparate Data Center locations that share the same IP address space and belong to the same VLAN (same VID).

LNV is one of the first approaches to implement LNV with Cumulus Linux, however, this is a method that cannot expand to multi-vendor scenarios and is intended to use as a client-server interaction between Spine and Leaves.

On the other hand, EVPN, can evolve to connect to MPLS/Segment-Routing infrastructures using the RFC 4364 MP-BGP protocol, which make this solution the better one. In fact, using VXLAN-EVPN is the heart of the SD-Access solution developed by Cisco Systems.

Modern Cumulus Linux NOS only supports EVPN.

Cumulus Linux, as part of the Open Networking paradigm, is one of the most flexible, scalable and high-performance NOS, in fact, it is named as Visionary in the 2020 Gartner Magic Quadrant for Data Center and Cloud Networking and will solve many problems the vendor-specific solutions are facing.

About security concerns, in [8], suggests using double encapsulation with IPSec and VXLAN, with that, solving the problem of sending traffic in plain text, but using special mechanisms to avoid fragmentation.

Acknowledgment

The authors would like to thank Mr. Edward Kelley for his help during the Cumulus Linux emulation phase.

References

[1] E. F. Naranjo, and G. D. Salazar Ch. “Underlay and overlay networks: The approach to solve addressing and segmentation problems in the new networking era: VXLAN encapsulation with Cisco and open source networks”. In 2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM) (pp. 1-6). IEEE. October, 2017.

[2] J. F. Chafla, S. Silva, A. Beltran, and H. Barba. “Tendencias tecnológicas para la modernización de centro de datos y propuesta de una arquitectura de data center moderno”. *RECIMUNDO:* Revista Científica de la Investigación y el Conocimiento, 2(1), 3-30. 2018

[3] “SD-Access (Fabric Network, Automation and Analytics LAN) – Campus Networks”, https://www.routexp.com/2020/03/sd-access-fabric-network-automation-and.html. (Accessed: Jul, 15, 2022).

[4] “Qué es la Arquitectura Spine-Leaf”, Aruba Networks, https://www.arubanetworks.com/es/faq/que-es-la-arquitectura-spine-leaf/. (Accessed: Jul 1, 2022).

[5] G. Salazar-Chacón, E. Naranjo, and L. Marrone. "Open networking programmability for VXLAN Data Centre infrastructures: Ansible and Cumulus Linux feasibility study." *Revista Ibérica de Sistemas e Tecnologias de Informação* E32 (2020): 469-482.

[6] “VXLAN an OTV – What is the difference?”, https://www.routexp.com/2018/03/vxlan-and-otv-what-is-difference.html. (Accessed: Jul 2, 2022).

[7] “VXLAN Network Architecture”, Huawei VXLAN support guide. https://support.huawei.com/enterprise/fr/doc/EDOC1000178188/5f42f6b3/vxlan-network-architecture. (Accessed: Jul 5, 2022).

[8] G. D Salazar-Chacón. “VXLAN-IPSec Dual-Overlay as a Security Technique in Virtualized Datacenter Environments”. In *2020 IEEE ANDESCON* (pp. 1-6). IEEE. October, 2020.

[9] “Juniper Expands Campus Portfolio”, SDN/SDWAN Juniper. https://www.channelfutures.com/sdn-sd-wan/juniper-networks-expands-campus-portfolio. (Accessed: Jul 2, 2022).

[10] “Networking Docs”. Cumulus Linux. https://docs.nvidia.com/networking-ethernet-software/cumulus-linux-51/Network-Virtualization/Ethernet-Virtual-Private-Network-EVPN/ (Accessed: Jul 1, 2022)

[11] G. D. Salazar Chacón. “Hybrid Networking SDN y SD-WAN: Interoperabilidad de arquitecturas de redes tradicionales y redes definidas por software en la era de la digitalización” (Doctoral dissertation, Universidad Nacional de La Plata). 2021

[12] Salazar-Chacón, G. D., & Marrone, L. (2020, November). OpenSDN Southbound Traffic Characterization: Proof-of-Concept Virtualized SDN-Infrastructure. In *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)* (pp. 0282-0287). IEEE.

[13] “Orionvm builds breakthrough IAAS Cloud Platform with Cumulus Linux”, https://www.orionvm.com/case-studies/cumulus-networks-case-study/ (Accessed: Jul 1, 2022).

[14] Andrade-Salinas, G., Salazar-Chacon, G., & Vintimilla, L. M. (2019, December). Integration of IoT Equipment as Transactional Endorsing Peers over a Hyperledger-Fabric Blockchain Network: Feasibility Study. In *International Conference on Applied Technologies* (pp. 95-109). Springer, Cham.