Lab 25.1: Understanding VXLAN – Overlay Networking Concepts

Companion to Chapter 25 – "VXLAN"

Objective

Understand how **VXLAN extends Layer 2 over Layer 3** by analyzing encapsulation behavior—using **free tools only**. Due to platform limitations, this lab focuses on **concept validation via Wireshark** and **Linux-based simulation**, not full VTEP configuration.

Important Note (2026):
True VXLAN requires vendor-specific features (Cisco NVE, Arista VXLAN, etc.). These images are **not freely available** for GNS3. Instead, we use **open methods** to observe VXLAN principles—preparing you for real implementations later.

Estimated Time

60–90 minutes

Required Tools

- **Wireshark (**free**)**
- **GNS3 (**free**)**
- **Linux appliance (**e.g., Ubuntu Cloud Image, free**)**
- **Optional**: Pre-captured VXLAN PCAP file (provided in book resources)

Topology Overview

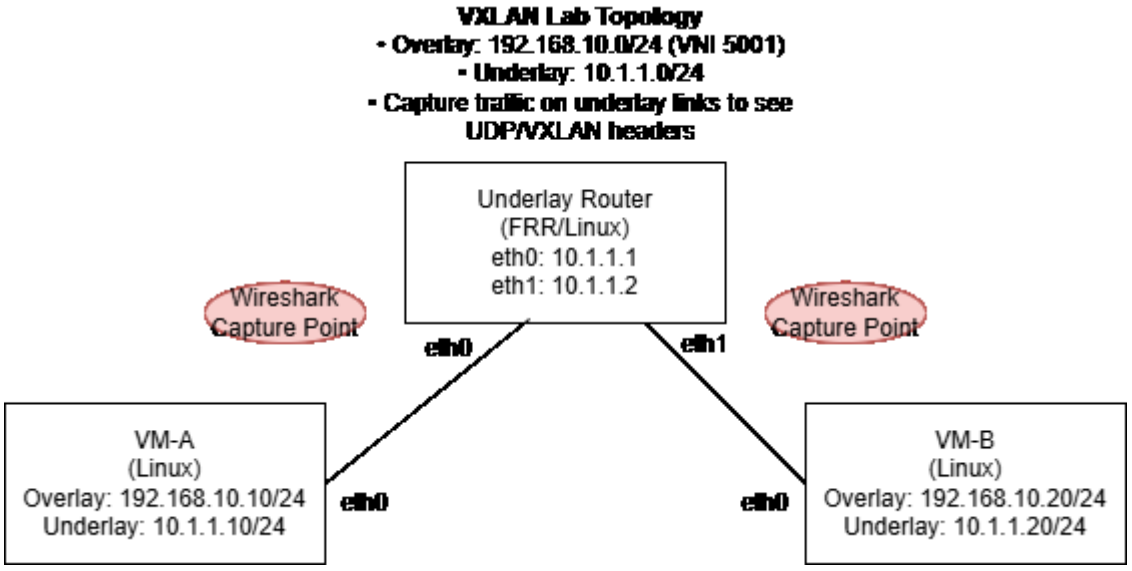

Figure L25.1: Lab 25.1 topology showing two Linux VMs (VM-A and VM-B) connected over an IP underlay network, simulating VXLAN overlay communication within the same VNI (5001).

- **VM-A and VM-B:** Linux virtual machines acting as end hosts in the overlay network (192.168.10.0/24)
- **Underlay Router:** Provides routed connectivity between VMs using IP addresses in the 10.1.1.0/24 segment
- **VXLAN Tunnel:** Created dynamically via Linux kernel VXLAN interfaces (VNI 5001), encapsulated in UDP (port 4789)
- **Wireshark Capture Points:** Located on underlay links to observe VXLAN packet structure

This visual helps students connect abstract VXLAN concepts—like encapsulation, VNI, and underlay/overlay separation—to a concrete, reproducible architecture using only free and open tools.

Step-by-Step Instructions

Part 1: Build a Simplified Topology

1. In GNS3, create:
    - **2x Linux VMs** (as “VM-A” and “VM-B”)
    - **1x Router** (FRRouting or simple Linux host as underlay)
2. Connect both VMs to the router over separate links
3. Assign underlay IPs:
    - **VM-A:** 10.1.1.10/24
    - **VM-B:** 10.1.1.20/24

Part 2: Configure the IP Underlay

On **VM-A:**

```
sudo ip link add vxlan0 type vxlan id 5001 dev eth0 dstport 4789

sudo ip addr add 192.168.10.10/24 dev vxlan0

sudo ip link set up dev vxlan0
```

On **VM-B:**

```
sudo ip link add vxlan0 type vxlan id 5001 dev eth0 dstport 4789

sudo ip addr add 192.168.10.20/24 dev vxlan0

sudo ip link set up dev vxlan0
```

This creates a **point-to-point VXLAN tunnel** using Linux kernel support (available in most modern distros).

Part 3: Test & Analyze

1. From VM-A:
    - Ping 192.168.10.20
2. In GNS3, **capture traffic** on the underlay link (eth0)
3. Open in **Wireshark** and observe:
    - **Outer IP header:** 10.1.1.10 → 10.1.1.20
    - **UDP header:** port **4789**
    - **VXLAN header: VNI = 5001**
    - **Inner Ethernet frame:** 192.168.10.10 → 192.168.10.20

Lab Report Questions

1. Why is the underlay network required for VXLAN?
2. What UDP port does VXLAN use by default?
3. How many bits are in a VNI? How many segments does that allow?
4. Why can't we use traditional VLANs at scale in cloud data centers?
5. In a real Egyptian cloud provider, why might you avoid multicast and use **EVPN + BGP** instead?

Key Takeaways

- VXLAN uses **MAC-in-UDP** encapsulation over IP underlay
- **VNI (24-bit)** replaces VLAN ID (12-bit) → **16 million segments**
- Real deployments use **EVPN** for control plane (not multicast)
- You don't need Cisco gear to **understand the protocol**

Real-World Connection

At a **Riyadh cloud startup**, engineers first learned VXLAN using **Linux and Wireshark**—just like this lab—before deploying it on Cisco ACI. Understanding the **packet structure** made troubleshooting effortless.

Instructor Notes

- **Common issue:** Students expect full GUI VTEP config—emphasize this is a **protocol-level lab**
- **Alternative:** Provide a **pre-recorded PCAP file** if Linux setup is too advanced
- **Extension:** Compare VXLAN vs. NVGRE vs. Geneve headers in Wireshark

- **Validation:** Use Appendix S: Overlay Networking Checklist

For deeper study, consider **Cisco Modeling Labs (CML)**—the only legal way to run Cisco VXLAN features.

Final Thought

You don’t need expensive licenses to understand the future of networking.
With **curiosity and open tools**, you can dissect the protocols powering NEOM, AWS, and Azure.