Trinet Result

The goal of this project is to design and deploy a simple network topology consisting of three routers in form of a triangle as backbone, with two different hosts placed on separate LANs. The goal is to establish IP connectivity between the hosts and verify that routing across the path work as expected. OSPF is chosen as routing protocol of this project.

Tools:

The tools used in this project are consisted of:

- Netlab: framework for defining and deploying network topologies.
- KVM + QEMU: The Hypervisor and the Emulator to run the VMs.
- Vagrant: Tool to create the VMs
- Libvirt: Management tool for managing the VMs.
- ContainerLab: CLI for creating and managing our custom network topology.
- Ansible: Configuration management tool to manage the VMs.

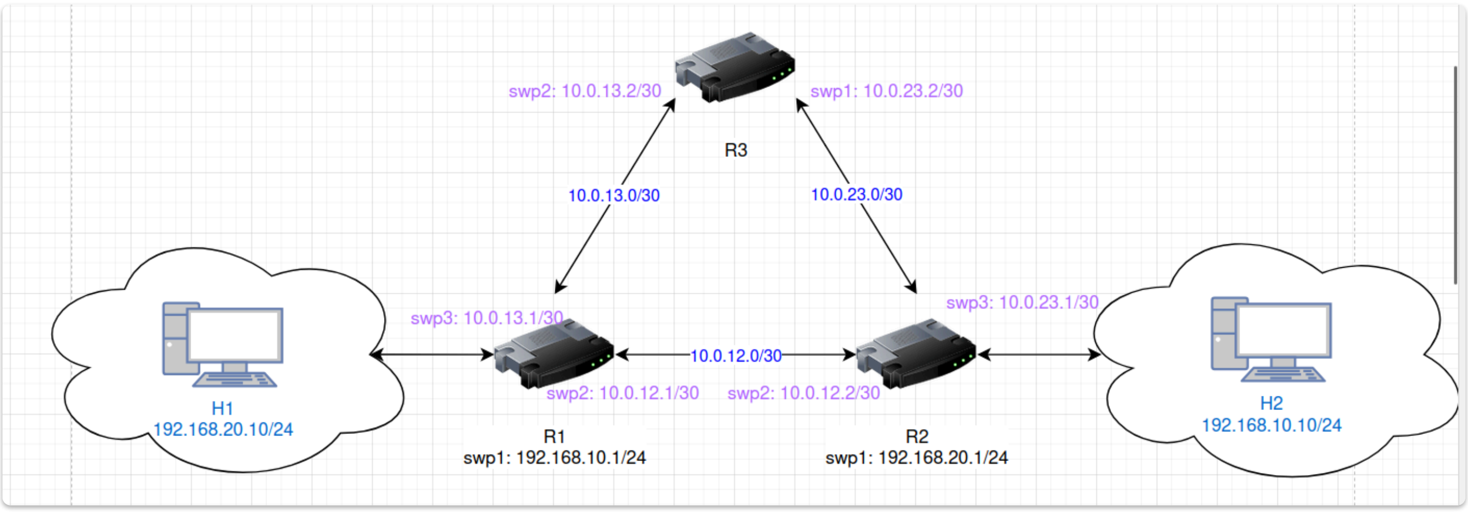
A Python virtual environment named `trinet` was created, and Netlab was installed inside it. After that, all the required dependencies were then installed using:

```
netlab install
```

Topology

The network is defined in a `topology.yml` file, which describes all the nodes, interfaces, Assigned IP addresses, and routing protocol. The topology includes five VMs which are:

- Three routers (R1, R2, R3) – running Cumulus VX.
- Two hosts (H1, H2) – running Ubuntu 24.04.



The contents of `topology.yml`:

```
# Default setting for the scenario.
defaults:
```

```yaml
  device: cumulus
  #provider: virtualbox
  provider: libvirt
  vagrant:
    memory: 1024
    cpus: 1

# Routing Protocol
module: [ospf]

# Routers are Cumulus VX
nodes:
  r1: {}
  r2: {}
  r3: {}
  # Two hosts, each on their corresponding LAN.
  h1: { device: linux }
  h2: { device: linux }

links:
  # Left host LAN: h1 --- r1
  - name: h1-lan
    h1:
      ipv4: 192.168.10.10/24
    r1:
      ipv4: 192.168.10.1/24
    ospf:
      area: 0
      passive: true

  # Right host LAN: h2 --- r2
  - name: h2-lan
    h2:
      ipv4: 192.168.20.10/24
    r2:
      ipv4: 192.168.20.1/24
    ospf:
      area: 0
      passive: true

  # Core triangle between routers
  - r1:
      ipv4: 10.0.12.1/30
    r2:
      ipv4: 10.0.12.2/30
    ospf:
      area: 0

  - r2:
      ipv4: 10.0.23.1/30
    r3:
      ipv4: 10.0.23.2/30
    ospf:
      area: 0

  - r1:
      ipv4: 10.0.13.1/30
    r3:
      ipv4: 10.0.13.2/30
    ospf:
      area: 0
```

Deployment Process

Running `netlab up` does the following actions:

1. Generates Vagrant files for our Libvirt.
2. Five VMs were created and booted (`R1`, `R2`, `R3`, `H1`, `H2`).
3. IP addresses were assigned to their corresponding interfaces.
4. OSPF was configured automatically on the routers.

Routing

By default, the hosts have the default gateway pointing to the `libvirt NAT router` (e.g: 192.168.121.1 on eth0). This gateway should be replaced with the correct default gateway from the topology, so:

```
# On H1
sudo ip route replace default via 192.168.10.1 dev eth1
# On H2
sudo ip route replace default via 192.168.20.1 dev eth1
```

Testing

The connectivity between hosts was tested with `ping` and `tracepath` (`traceroute` is not installed).

From H1 (192.168.20.10) to H2 (192.168.10.10):

The outputs were:

```
ping -c 3 192.168.20.10
PING 192.168.20.10 (192.168.20.10) 56(84) bytes of data.
64 bytes from 192.168.20.10: icmp_seq=1 ttl=62 time=2.01 ms
64 bytes from 192.168.20.10: icmp_seq=2 ttl=62 time=1.89 ms
64 bytes from 192.168.20.10: icmp_seq=3 ttl=62 time=3.09 ms

--- 192.168.20.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.887/2.329/3.089/0.539 ms
vagrant@h1:~$ tracepath 192.168.20.10
 1?: [LOCALHOST]                      pmtu 9500
 1:  swp1.r1                                              0.596ms
 1:  swp1.r1                                              0.310ms
 2:  swp1.r1                                           0.155ms pmtu 1500
 2:  swp2.r2                                           0.545ms
 3:  h2                                                0.975ms reached
     Resume: pmtu 1500 hops 3 back 3
```

From H2 (192.168.10.10) to H1 (192.168.20.10):

The outputs were:

```
ping -c 3 192.168.10.10
PING 192.168.10.10 (192.168.10.10) 56(84) bytes of data.
64 bytes from 192.168.10.10: icmp_seq=1 ttl=62 time=1.59 ms
64 bytes from 192.168.10.10: icmp_seq=2 ttl=62 time=1.22 ms
64 bytes from 192.168.10.10: icmp_seq=3 ttl=62 time=2.36 ms

--- 192.168.10.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.217/1.724/2.363/0.476 ms
vagrant@h2:~$ tracepath 192.168.10.10
 1?: [LOCALHOST]                     pmtu 9500
 1:  swp1.r2                                           0.858ms
 1:  swp1.r2                                           0.285ms
 2:  swp1.r2                                           0.176ms pmtu 1500
 2:  swp2.r1                                           1.318ms
 3:  h1                                                1.352ms reached
     Resume: pmtu 1500 hops 3 back 3
```

Routing Through R3

To test the routing also from `R3` (Redundancy) the direct link between R1 and R2 was disabled:

```
sudo ip link set swp2 down
```

This leads to OSPF protocol recalculating the traffic between H1 and H2 and passing the traffic through R3. Again, The connectivity between hosts was tested with `ping` and `tracepath`.

On H1:

```
ping -c 3 192.168.20.10
PING 192.168.20.10 (192.168.20.10) 56(84) bytes of data.
64 bytes from 192.168.20.10: icmp_seq=1 ttl=61 time=3.64 ms
64 bytes from 192.168.20.10: icmp_seq=2 ttl=61 time=2.06 ms
64 bytes from 192.168.20.10: icmp_seq=3 ttl=61 time=5.44 ms

--- 192.168.20.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 2.063/3.716/5.443/1.380 ms
vagrant@h1:~$ tracepath 192.168.20.10
 1?: [LOCALHOST]                     pmtu 9500
 1:  swp1.r1                                           0.344ms
 1:  swp1.r1                                           0.255ms
 2:  swp1.r1                                                 0.155ms pmtu 1500
 2:  swp2.r3                                                 0.697ms
 3:  swp3.r2                                                 0.870ms
 4:  h2                                                      0.925ms reached
     Resume: pmtu 1500 hops 4 back 4
```

Conclusion

The Trinet project successfully demonstrated how to build and operate a small routed network using Cumulus VX routers, OSPF, and Netlab’s automation stack.