CO515: Advances in Computer Networks: Selected Topics – Lab09
Simulating a Data Center Network environment

E/19/446: Wijerathna I.M.K.D.I.

13/06/2024

Activity 1: Understanding Data Centre Network Topologies

1. Diagrams of 3-tier, spine-leaf, and mesh topologies

1.1) 3-tier topology

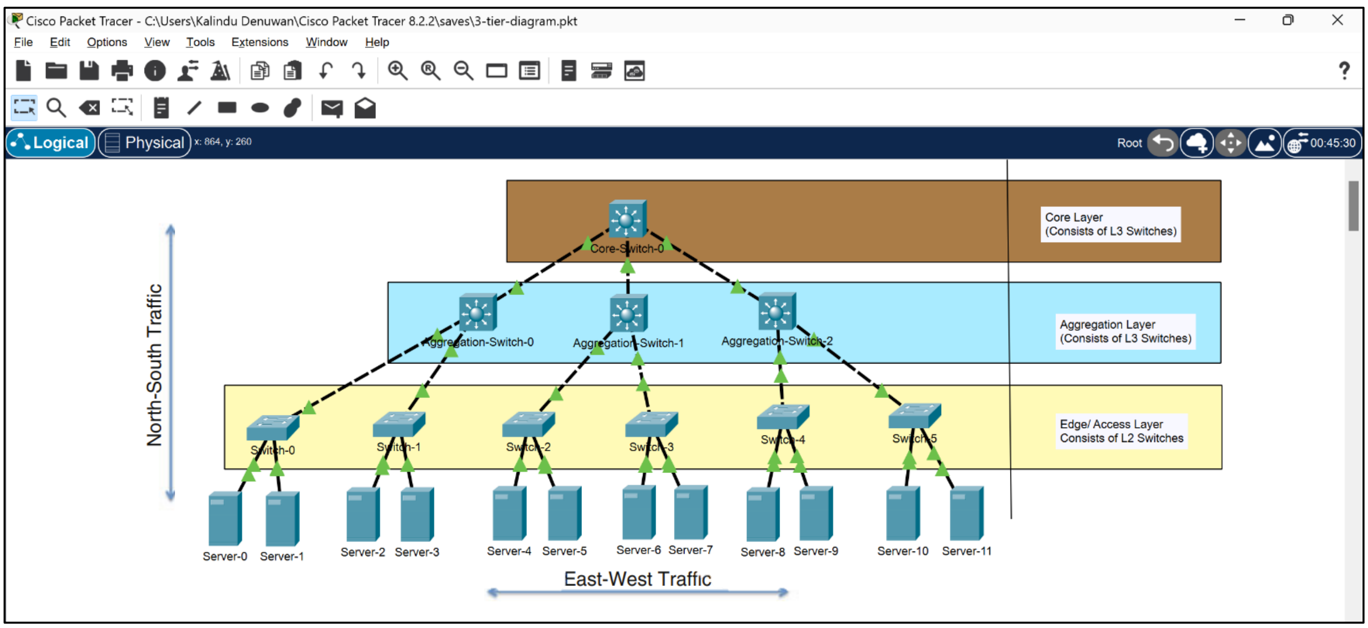


Figure 1: 3-Tier Topology Diagram

- *Structure: Consists of three layers: core, aggregation, and edge(access) layers.*
- *Traffic Flow: Traffic typically flows from edge switches to aggregation switches, and then to core switches. Higher-tier switches must be high-performance and reliable to support communication among many servers, with server numbers limited by switch port availability.*
- *Scalability: Scalable horizontally by adding more edge switches or vertically by upgrading core switches (adding another core switch layer).*

The above drawn topology is a basic three-tier topology and based on this topology there have been developed many variations of this topology later on like fat-tree 3-tier topology etc.

1.2) Spine-leaf topology

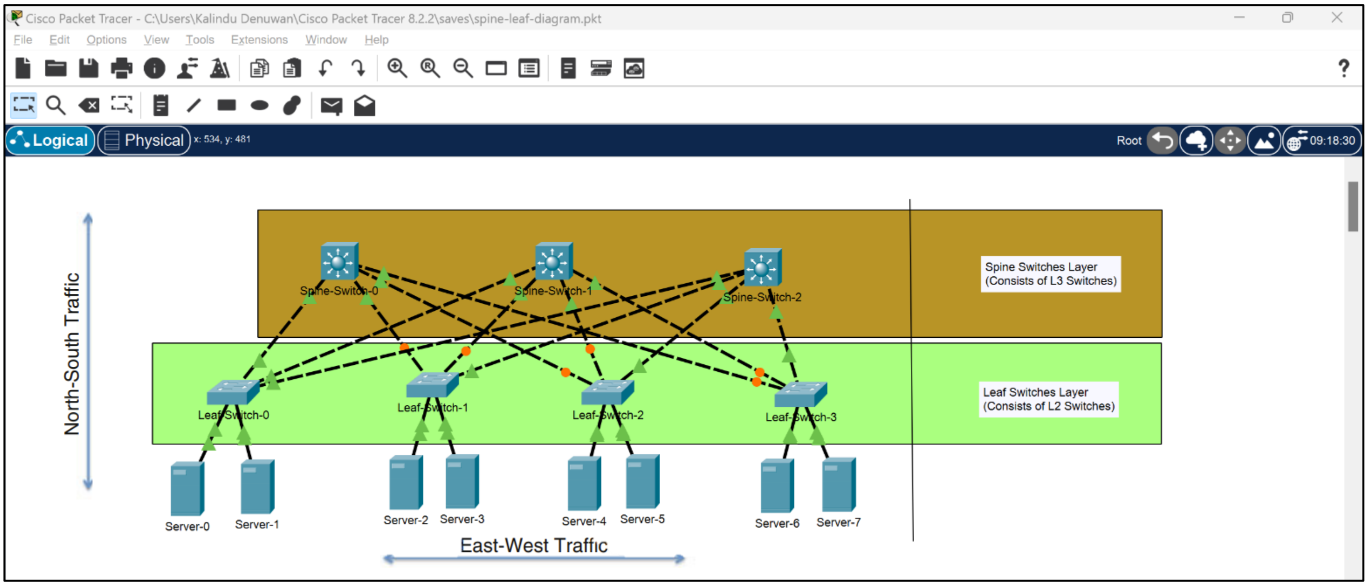


Figure 2: Spine Leaf Topology Diagram

- *Structure: Consists of two layers: spine switches and leaf switches.*
- *Traffic Flow: Leaf switches connect to servers and end-devices, while spine switches provide interconnection between leaf switches.*
- *Performance: Spine Leaf topology Offers high bandwidth and low latency due to non-blocking architecture.*

1.3) Mesh topology

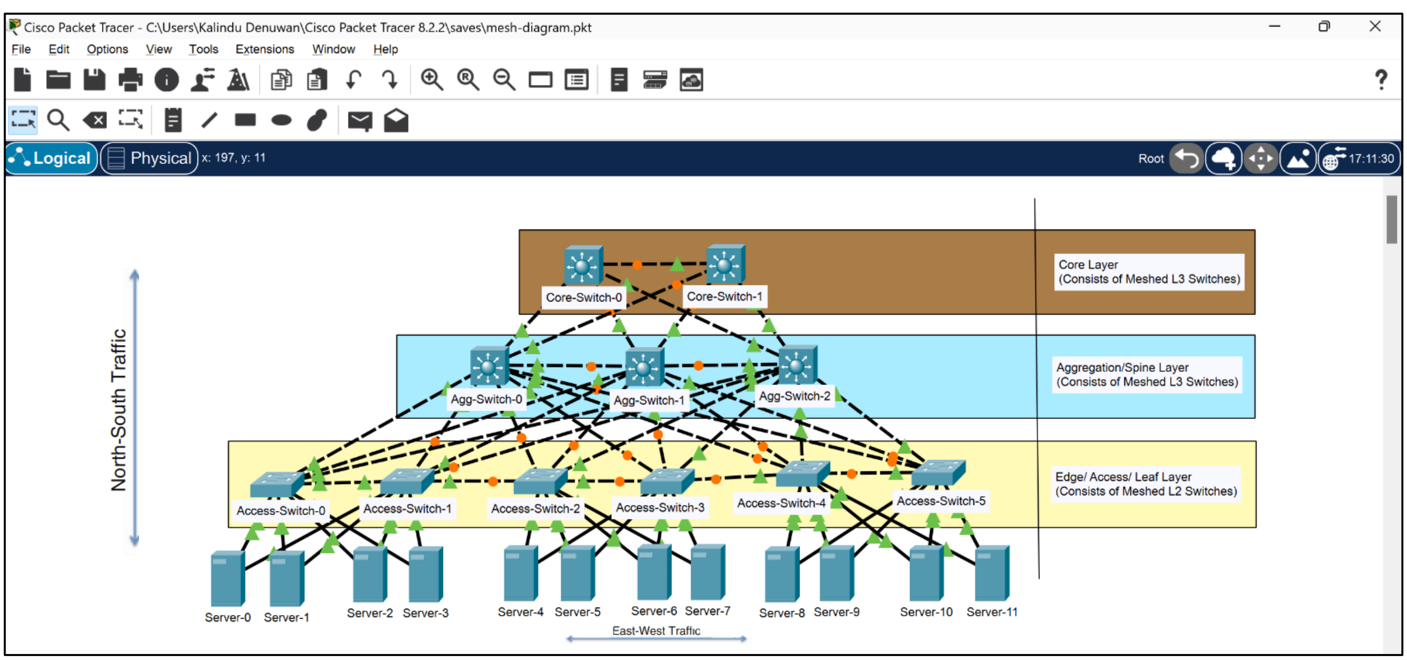


Figure 3: Mesh Topology Diagram

- *Structure: Each device is interconnected with every other device in the network (directly connected to adjacent devices in the same layer and all devices in adjacent layers(up/down)).*
- *Redundancy: Provides high redundancy and fault tolerance as multiple paths are available between any two devices.*
- *Complexity: More complex to design and manage due to numerous interconnections, but offers robust connectivity and flexibility.*

2. A comparison table of the topologies

Comparison of 3-tier, spine-leaf and mesh topologies in data center networks.

Characteristic	3-Tier Topology	Spine-Leaf Topology	Mesh Topology
Structure	Core layer, aggregation (a.k.a distribution) layer, and edge (a.k.a access) layer.	Spine switches and leaf switches.	Fully interconnected, every device connects to every other device.
Traffic Flow	Edge switches connect to aggregation switches, then to core switches.	Leaf switches connect directly to servers/devices. Spine switches provide interconnection between leaf switches.	Directly between any two devices, multiple paths available.
Redundancy	Limited redundancy due to hierarchical design.	High redundancy with non-blocking architecture.	High redundancy with multiple paths between devices.
Scalability	Scalable vertically (upgrading core switches) and horizontally (adding more access switches).	Scalable horizontally with more leaf-spine pairs.	Scalable, but complexity increases with the number of devices.
Latency	Medium to low latency compared to mesh topology.	Low latency due to direct connections and non-blocking switches.	Low latency due to direct connections; can vary based on path choice.

Complexity	Moderate complexity in design and management.	Moderate complexity; simpler than mesh but more complex than 3-tier.	High complexity due to numerous connections; challenging to manage.
Fault Tolerance	Limited fault isolation; entire tier can be affected by a failure.	Good fault isolation; failure in one leaf-spine pair does not affect others.	Excellent fault tolerance; failures in one connection do not affect overall connectivity.
Common Use Cases	Traditional enterprise networks and some data centers.	Modern data centers which requires high bandwidth and low latency.	High-performance computing, research networks, and specific applications requiring robust connectivity.

Activity 2: Exploring Data Centre Architecture

1. Description of each layer's functions

In a typical data center architecture, the network is usually organized into three layers: core, aggregation (or distribution), and access. Each layer has specific functions and responsibilities, contributing to the overall performance, scalability, and reliability of the data center network.

1. Core Layer

- *Purpose: The core layer is the backbone of the data center network. It provides high-speed and highly resilient connectivity between different parts of the data center and beyond, to other data centers and the internet.*
- *Runs interior routing protocols like OSPF or ISIS and balances traffic between the core and aggregation layers.*
- *Functions:*
  - *High-speed Switching: Facilitates rapid and efficient data transfer across the network.*
  - *Redundancy and Fault Tolerance: Ensures minimal downtime and data loss through redundancy, often employing techniques like load balancing and failover.*
  - *Scalability: Supports large volumes of data traffic and can be easily scaled to accommodate growing network demands.*

2. Aggregation (Distribution) Layer

- *Purpose: The aggregation layer, also known as the distribution layer, serves as an intermediary between the core and access layers. It consolidates data from multiple access switches before transmitting it to the core layer.*
- *Offers redundancy functions and merges multiple Layer 2 domains to prevent spanning tree processing issues.*
- *an integrate service modules and use devices like firewalls and server load balancers to optimize and secure application flows.*

- *Functions:*
    - *Traffic Aggregation: Combines data traffic from multiple access layer switches, reducing the number of connections to the core layer.*
    - *Policy Implementation: Enforces network policies such as security, routing, and quality of service (QoS).*
    - *Load Balancing: Distributes traffic evenly across the network to optimize performance and prevent bottlenecks.*

3. Access Layer

- *Purpose: The access layer is the most peripheral layer, directly connecting end devices, such as servers and mainframes, to the network.*

- *Functions:*
    - *Device Connectivity: Provides network access to servers, storage devices, and other end-user devices.*
    - *Network Access Control: Implements security measures like port security, VLAN segmentation, and access control lists (ACLs) to protect the network.*
    - *Data Distribution: Facilitates the initial data transfer from end devices to higher network layers.*

2. Data Centre architecture diagram

The following architecture was created based on Cisco's recommended DCN topology
Please Turn Over.

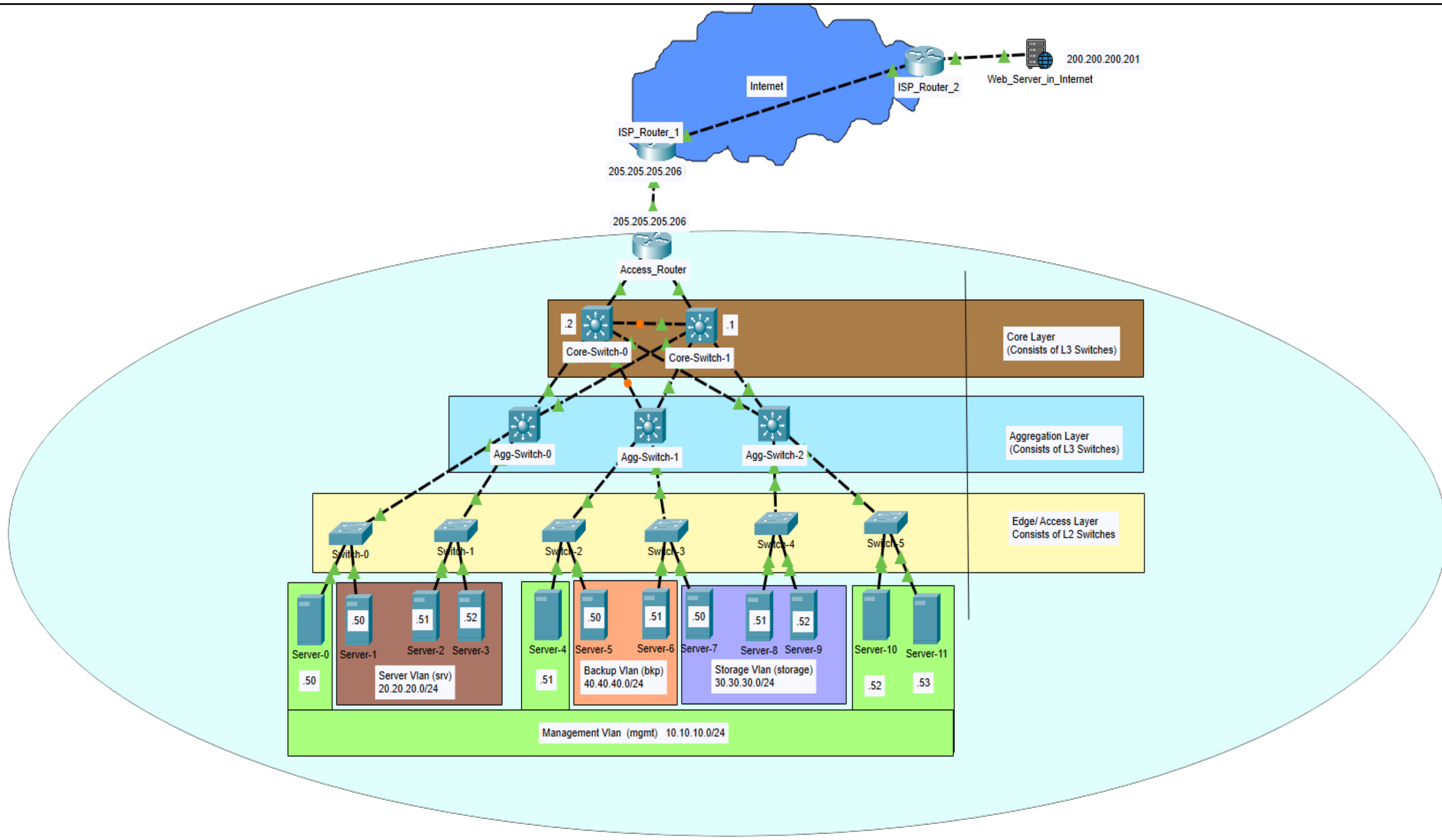


Figure 4: Data Center Network Implemented in Cisco Packet Tracer

Activity 3: Basic Network Configuration

1.Screenshots of configuration commands, results and a brief report on the configuration steps and outcomes

Report on Data Center Network Configuration

The data center network was implemented in cisco-packet tracer and the complete network diagram is as provided in the previous page as Figure-4.

Architecture

The data center network is designed using a basic three-tier architecture comprising access, aggregation, and core layers.

Servers and VLANs

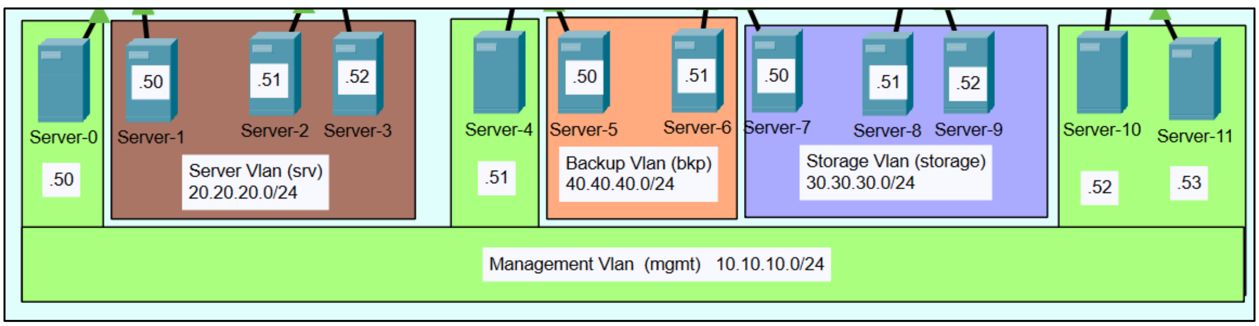


Figure 5:Server and VLAN config

There are 12 servers in total (server-0 to server-11).

- *VLAN 10: mgmt (Management, IP range: 10.10.10.0/24)*
  - *Servers: server-0, server-4, server-10, server-11*
- *VLAN 20: srv (Servers, IP range: 20.20.20.0/24)*
  - *Servers: server-1, server-2, server-3*
- *VLAN 30: storage (Storage, IP range: 30.30.30.0/24)*
  - *Servers: server-7, server-8, server-9*
- *VLAN 40: bkp (Backup, IP range: 40.40.40.0/24)*
  - *Servers: server-5, server-6*

Access Layer and Assigning Ports for each VLAN

6x L2 switches: Each switch connects to two servers, with VLANs configured as per server assignments.

- *Switch-0: VLAN 10 and VLAN 20 configured*
- *Switch-1: VLAN 20 configured*
- *Switch-2: VLAN 10 and VLAN 40 configured*
- *Switch-3: VLAN 40 and VLAN 30 configured*
- *Switch-4: VLAN 30 configured*

- *Switch-5: VLAN 10 configured*

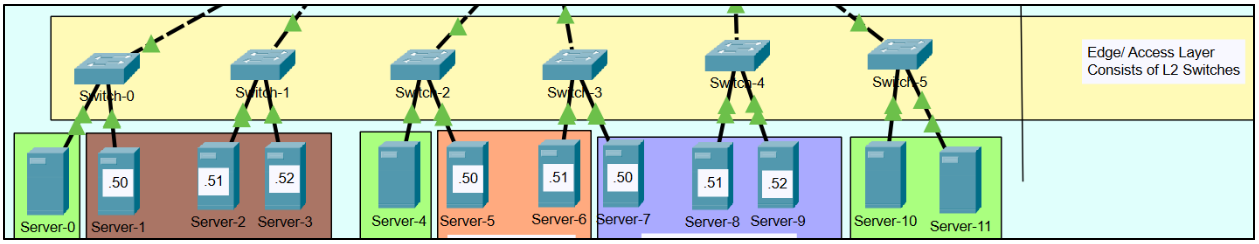


Figure 6: Assigning Ports for each VLAN in access layer switches

Above VLAN configurations with their verification commands:

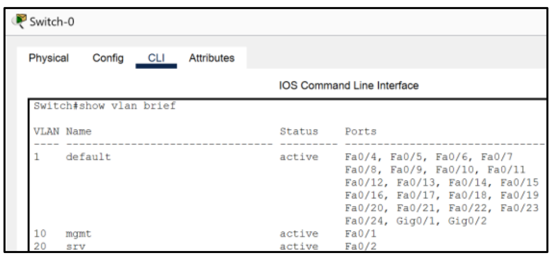


Figure 8: Access-Switch-0

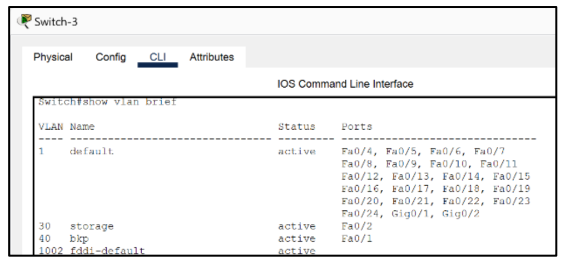


Figure 7: Access-Switch-3

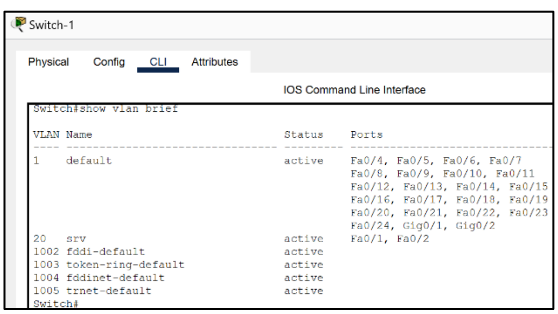


Figure 10: Access-Switch-1

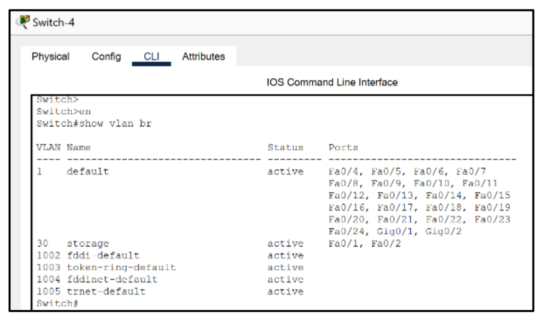


Figure 9: Access-Switch-4

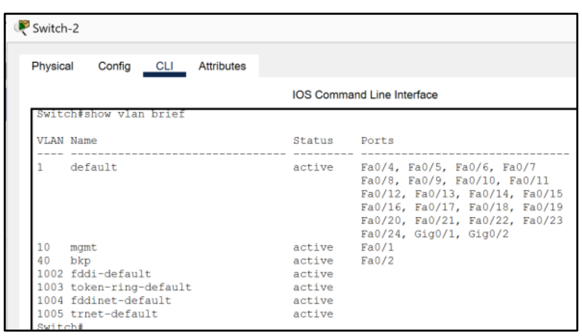


Figure 11: Access-Switch-2

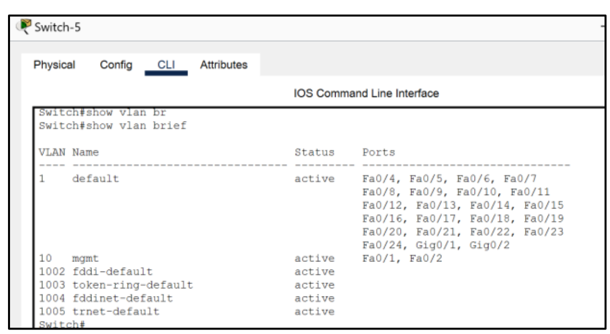


Figure 12: Access-Switch-5

__Aggregation Layer__

3x L3 switches

- *Each aggregation switch connects to 2 access switches.*
- *Trunk ports are configured on each aggregation switch to allow inter-VLAN communication.*
- *All four ports on each aggregation switch are trunked in this specific scenario.*
- *At the moment IP routing and other l3 features haven't been specifically used by these switches in my setup, however those technologies can be effectively used in configuring load balancing on aggregate layer switches.*

  Following is the VLAN configuration of aggregate-switch-0. Other 2 agg-switches are similar.

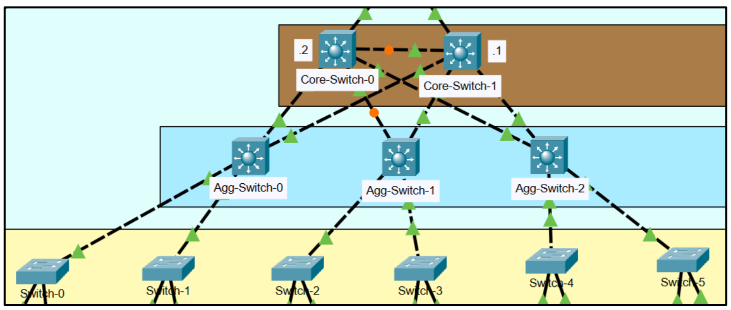
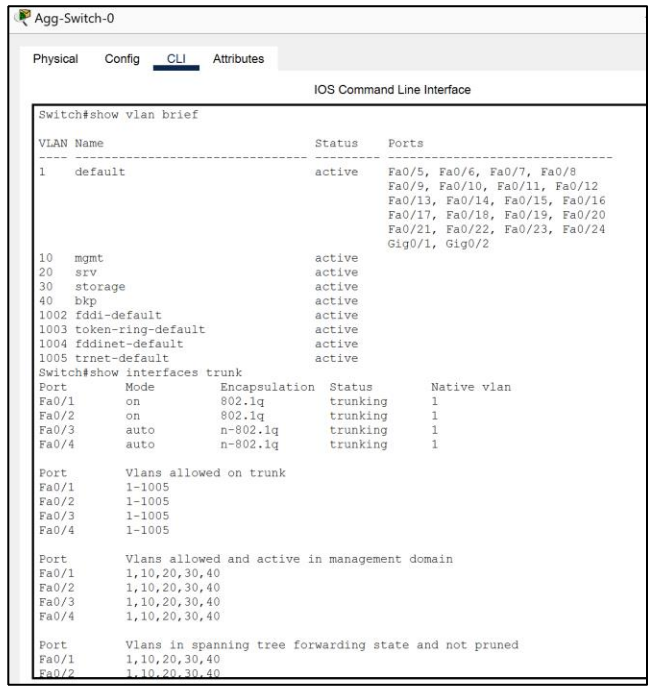


Figure 14: Aggregate Switches connections



Agg-Switch-0

Physical Config CLI Attributes

IOS Command Line Interface

```
Switch#show vlan brief

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Fa0/5, Fa0/6, Fa0/7, Fa0/8
                                                Fa0/9, Fa0/10, Fa0/11, Fa0/12
                                                Fa0/13, Fa0/14, Fa0/15, Fa0/16
                                                Fa0/17, Fa0/18, Fa0/19, Fa0/20
                                                Fa0/21, Fa0/22, Fa0/23, Fa0/24
                                                Gig0/1, Gig0/2
10   mgmt                             active
20   srv                              active
30   storage                          active
40   bkp                              active
1002 fddi-default                     active
1003 token-ring-default               active
1004 fddinet-default                  active
1005 trnet-default                    active
Switch#show interfaces trunk
Port        Mode         Encapsulation  Status        Native vlan
Fa0/1       on           802.1q         trunking      1
Fa0/2       on           802.1q         trunking      1
Fa0/3       auto         n-802.1q       trunking      1
Fa0/4       auto         n-802.1q       trunking      1

Port        Vlans allowed on trunk
Fa0/1       1-1005
Fa0/2       1-1005
Fa0/3       1-1005
Fa0/4       1-1005

Port        Vlans allowed and active in management domain
Fa0/1       1,10,20,30,40
Fa0/2       1,10,20,30,40
Fa0/3       1,10,20,30,40
Fa0/4       1,10,20,30,40

Port        Vlans in spanning tree forwarding state and not pruned
Fa0/1       1,10,20,30,40
Fa0/2       1,10,20,30,40
```

Figure 13: VLAN allocations on an aggregate-switch

__Core Layer__

2x L3 switches

- *Core-Switch 1:*
  - *Default gateway for devices in VLAN 10 and VLAN 30*
  - *`interface vlan 10` => IP address 10.10.10.1*
  - *`interface vlan 30` => IP address 30.30.30.1*
- *Core-Switch 0:*
  - *Default gateway for VLAN 20 and VLAN 40*
  - *`interface vlan 20` => IP address 20.20.20.2*
  - *`interface vlan 40` => IP address 40.40.40.2*

Moreover, OSPF have been used to route traffic from core-switches to the access router.

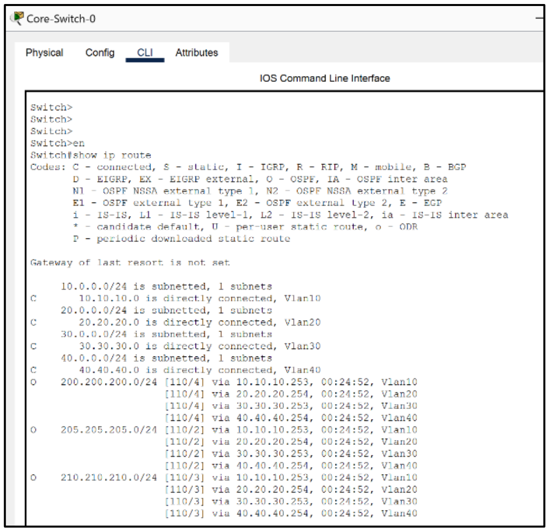


Figure 16: Core-Switch-0 Routing table with OSPF

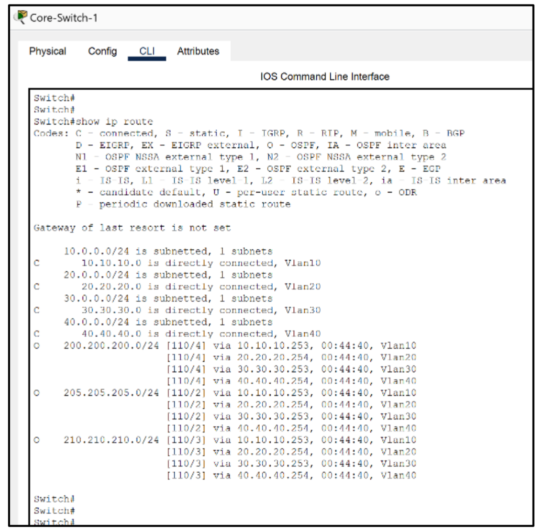


Figure 15: Core-Switch-1 Routing Table with OSPF

<u>Access Router Configuration</u>

Connected to the core switches through sub-interfaces configured accordingly with dot1Q encapsulation enabled for VLANs.

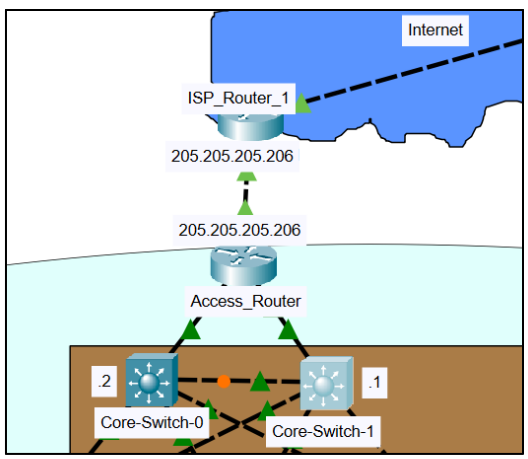


Figure 17: Access Router with it's internal and external connections

Internet Connection:

- *`205.205.205.206` is connected to an ISP router (ISP_Router_1) as shown above in Figure 17, providing internet access.*

Access_Router

Physical Config CLI Attributes

IOS Command Line Interface

```
Router#show ip interface br
Router#show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet0/0     unassigned      YES manual up                    up
GigabitEthernet0/0.10  unassigned      YES unset  up                    up
GigabitEthernet0/0.20  20.20.20.254    YES manual up                    up
GigabitEthernet0/0.30  unassigned      YES unset  up                    up
GigabitEthernet0/0.40  40.40.40.254    YES manual up                    up
GigabitEthernet0/1     unassigned      YES manual up                    up
GigabitEthernet0/1.10  10.10.10.253    YES manual up                    up
GigabitEthernet0/1.20  unassigned      YES unset  up                    up
GigabitEthernet0/1.30  30.30.30.253    YES manual up                    up
GigabitEthernet0/1.40  unassigned      YES unset  up                    up
GigabitEthernet0/2     205.205.205.206 YES manual up                    up
Vlan1                  unassigned      YES unset  administratively down down
Router#
```

Figure 18: IP Configuration in Access Router

Access_Router

Physical Config CLI Attributes

IOS C

```
Router#
Router#show IP route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       10.10.10.0/24 is directly connected, GigabitEthernet0/1.10
L       10.10.10.253/32 is directly connected, GigabitEthernet0/1.10
     20.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       20.20.20.0/24 is directly connected, GigabitEthernet0/0.20
L       20.20.20.254/32 is directly connected, GigabitEthernet0/0.20
     30.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       30.30.30.0/24 is directly connected, GigabitEthernet0/1.30
L       30.30.30.253/32 is directly connected, GigabitEthernet0/1.30
     40.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C       40.40.40.0/24 is directly connected, GigabitEthernet0/0.40
L       40.40.40.254/32 is directly connected, GigabitEthernet0/0.40
O    200.200.200.0/24 [110/3] via 205.205.205.205, 01:32:49, GigabitEthernet0/2
     205.205.205.0/24 is variably subnetted, 2 subnets, 2 masks
C       205.205.205.0/24 is directly connected, GigabitEthernet0/2
L       205.205.205.206/32 is directly connected, GigabitEthernet0/2
O    210.210.210.0/24 [110/2] via 205.205.205.205, 01:32:49, GigabitEthernet0/2

Router#
```

Figure 19: Routing Table of the Access Router

```
router ospf 1
 log-adjacency-changes
 network 205.205.205.0 0.0.0.255 area 0
 network 10.10.10.0 0.0.0.255 area 0
 network 20.20.20.0 0.0.0.255 area 0
 network 30.30.30.0 0.0.0.255 area 0
 network 40.40.40.0 0.0.0.255 area 0
!
ip classless
```

Figure 20: OSPF configuration at the Access Router

To demonstrate that the data center network is connected to the internet, I have used a OSPF configured network with a few routers of an ISP's network.

To demonstrate that the Data center network can properly communicate with the outside internet, I have created and linked to a HTTP web server with IP `200.200.200.201` as shown in the below Figure.

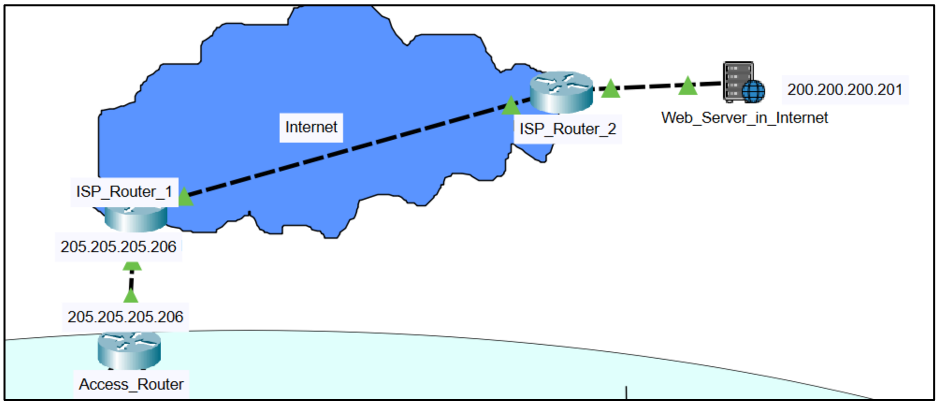


Figure 21: Demonstrating the DCN being connected to an external network (internet)

<u>*OSPF Routing Configuration*</u>

- *As shown in the above configurations and verification command screenshots, dynamic routing is configured using OSPF across all routers and core switches. A single OSPF area (`area 0`) has been used for ease of demonstration.*
- *This setup provides both inter-VLAN and inter-network routing.*

<u>*Verification and Testing*</u>

Throughout the setup, connectivity was verified using ping tests:

- *Between the internal network and external networks.*
- *Between different VLANs.*

Confirmed that the data center network is functioning as expected, with successful ping results demonstrating proper configuration and connectivity.

Between VLAN Communication:

Server-0 belongs to VLAN 10 (Management VLAN). Pinging from server-0 to other 3 VLANs.

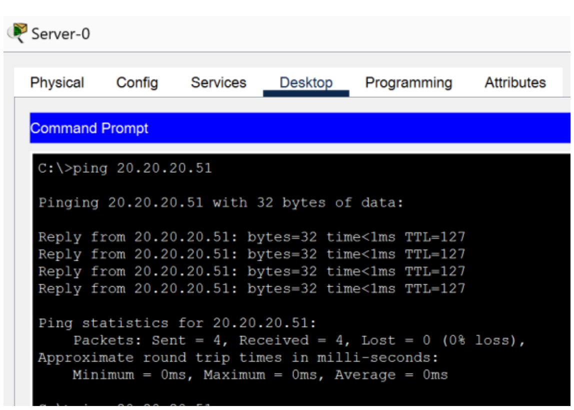


Figure 23: pinging from VLAN 10 to VLAN 20

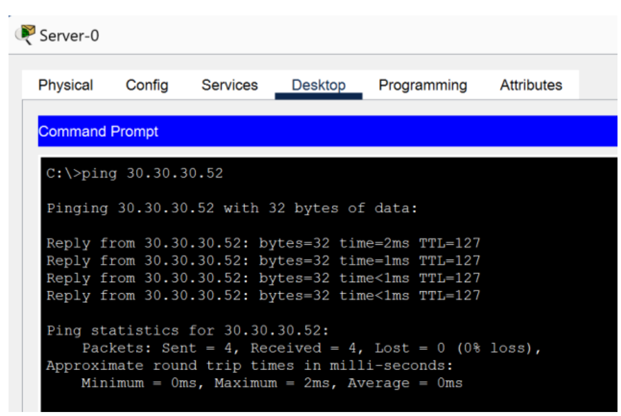


Figure 22: pinging from VLAN 10 to VLAN 30

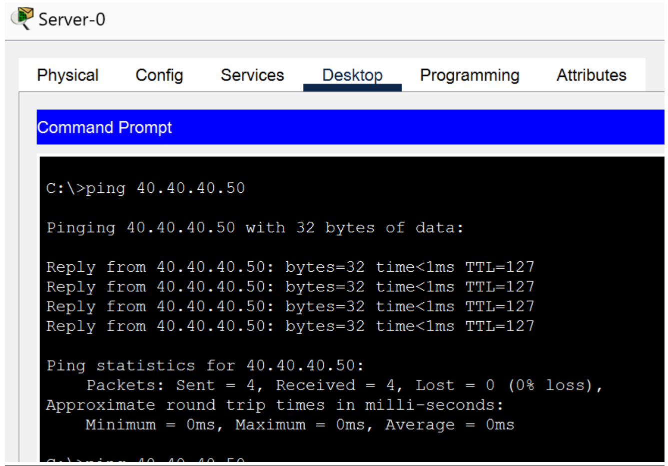


Figure 24: pinging from VLAN 10 to VLAN 40

Similarly, trying out with Server-6 which belongs to VLAN 40 (Backup VLAN). Pinging from server-6 to other 3 VLANs

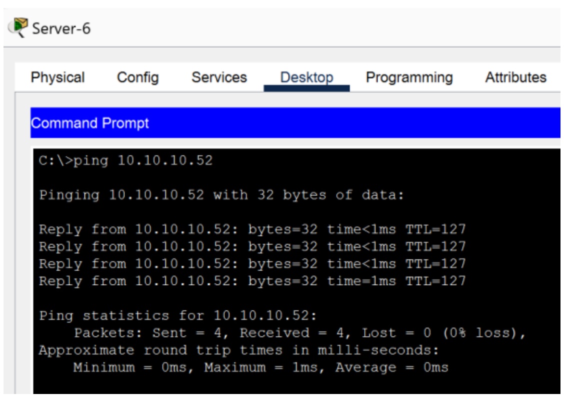


Figure 26: pinging from VLAN 40 to VLAN 10

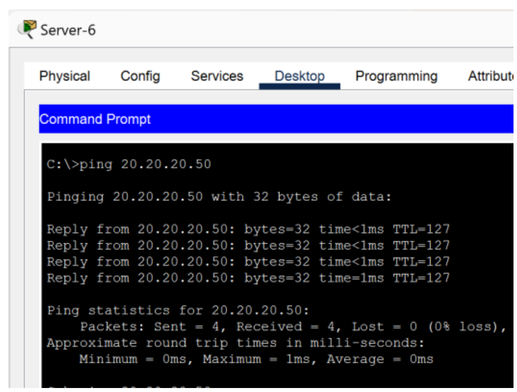


Figure 25: pinging from VLAN 40 to VLAN 20

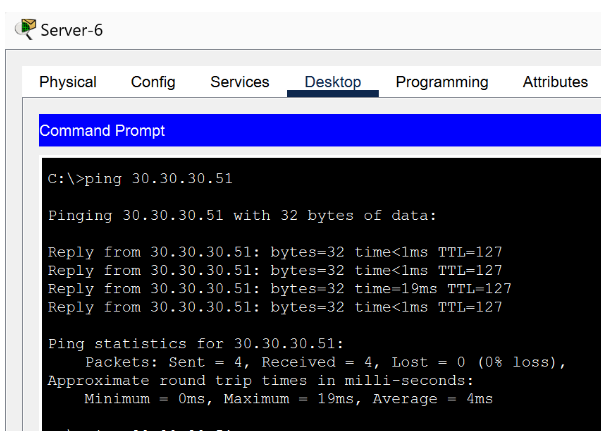


Figure 27: pinging from VLAN 40 to VLAN 30

Communication from Devices in Data Center Network to Devices in Internet:

For this, I have configured 2 HTTP servers

1. *In the internet with IP : 200.200.200.201*
2. *Inside the Data Center Network(server VLAN) with IP : 20.20.20.52*

Trying to access the WEB server in the internet from a DCN Server (server-10)

Figure 28: Web Server in the Internet accessed from the DCN

Trying to access the WEB server in the DCN from a Device in the Internet (internet-web-server)

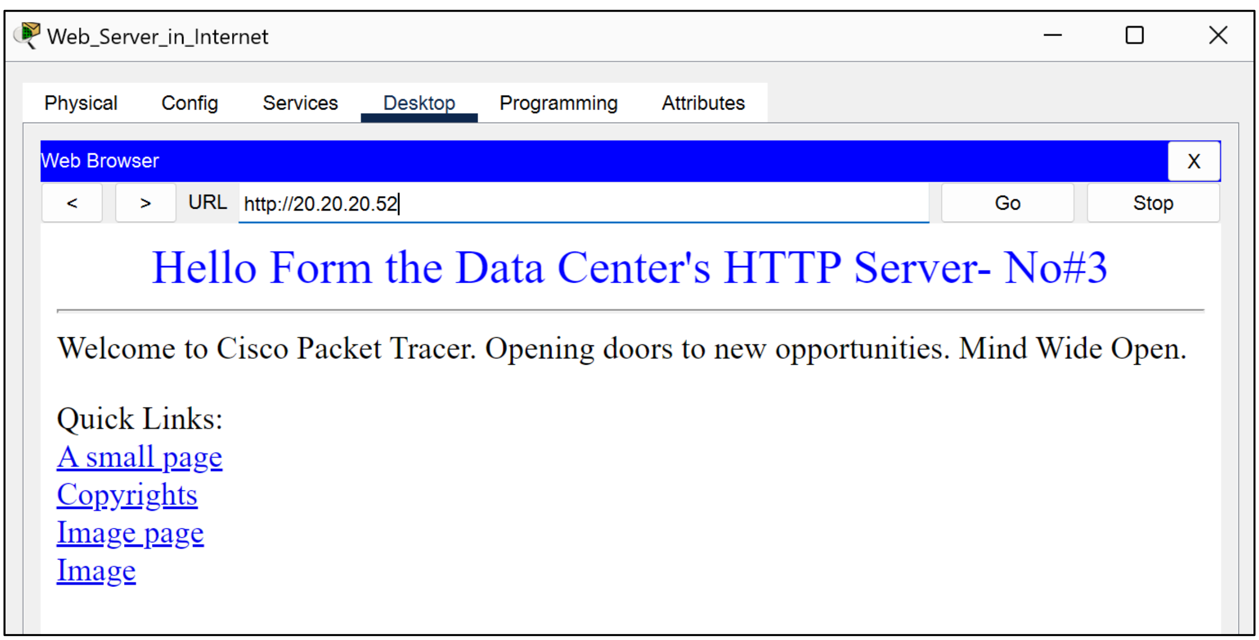


Figure 29: Web Server inside the DCN accessed from the Internet

Summary

The above configuration established a robust three-tier data center network, ensuring effective VLAN segmentation, inter-VLAN communication, and external network access through dynamic OSPF routing. The network supports efficient management, server, storage, and backup functions, demonstrating both internal and external connectivity.

Activity 4: Introduction to Network Virtualization in Data Center Networks

1. Diagram and virtual network configuration

The following Data Center Network is implemented using Mininet and Python. The diagram is based on the basic three tier architecture.

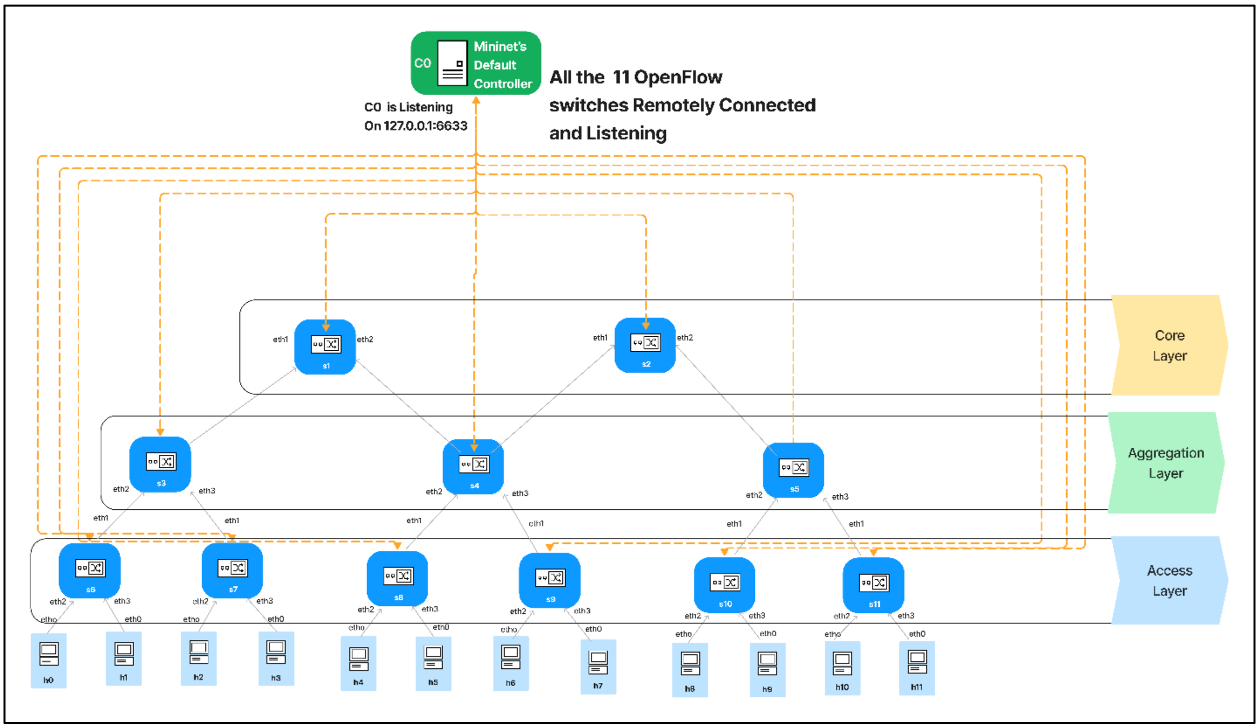


Figure 30: SDN with Data Center Network Diagram

Initially configured without inter-VLAN routing, hence obtained the following:

```
mininet> nodes
available nodes are:
c0 h0 h1 h10 h11 h2 h3 h4 h5 h6 h7 h8 h9 s1 s10 s11 s2 s3 s4 s5 s6 s7 s8 s9
mininet> pingall
*** Ping: testing ping reachability
h0 -> h1 h2 h3 X X X X X X X X
h1 -> h0 h2 h3 X X X X X X X X
h2 -> h0 h1 h3 X X X X X X X X
h3 -> h0 h1 h2 X X X X X X X X
h4 -> X X X X h5 h6 h7 X X X X
h5 -> X X X X h4 h6 h7 X X X X
h6 -> X X X X h4 h5 h7 X X X X
h7 -> X X X X h4 h5 h6 X X X X
h8 -> X X X X X X X X h9 h10 h11
h9 -> X X X X X X X X h8 h10 h11
h10 -> X X X X X X X X h8 h9 h11
h11 -> X X X X X X X X h8 h9 h10
*** Results: 72% dropped (36/132 received)
mininet>
```

Figure 31: Pingall for Mininet Configuration without Inter-VLAN Routing

Following is the Result Received after configuring inter-VLAN routing,

```
mininet> pingall
*** Ping: testing ping reachability
h0 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11
h1 -> h0 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11
h2 -> h0 h1 h3 h4 h5 h6 h7 h8 h9 h10 h11
h3 -> h0 h1 h2 h4 h5 h6 h7 h8 h9 h10 h11
h4 -> h0 h1 h2 h3 h5 h6 h7 h8 h9 h10 h11
h5 -> h0 h1 h2 h3 h4 h6 h7 h8 h9 h10 h11
h6 -> h0 h1 h2 h3 h4 h5 h7 h8 h9 h10 h11
h7 -> h0 h1 h2 h3 h4 h5 h6 h8 h9 h10 h11
h8 -> h0 h1 h2 h3 h4 h5 h6 h7 h9 h10 h11
h9 -> h0 h1 h2 h3 h4 h5 h6 h7 h8 h10 h11
h10 -> h0 h1 h2 h3 h4 h5 h6 h7 h8 h9 h11
h11 -> h0 h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Results: 0% dropped (132/132 received)
mininet>
```

Figure 32: Pingall after configuring inter-VLAN routing

Therefore, it can be observed that the Data center network has been configured properly with SDN. Here, I used the default SDN controller by Mininet.

The following is the python code used for this:

```python
from mininet.net import Mininet
from mininet.node import Controller, OVSSwitch
from mininet.cli import CLI
from mininet.link import TCLink
from mininet.log import setLogLevel, info

def createDataCenter():
    net = Mininet(controller=Controller, link=TCLink, switch=OVSSwitch)

    info('*** Adding controller\n')
    net.addController('c0')

    info('*** Adding switches\n')
    # Core switches
    info('*** ||||||||| Adding Core Layer....\n')
    core1 = net.addSwitch('s1')
    core2 = net.addSwitch('s2')

    # Aggregation switches
    info('*** ||||||||| Adding Aggregation Layer....\n')
    agg1 = net.addSwitch('s3')
    agg2 = net.addSwitch('s4')
    agg3 = net.addSwitch('s5')

    # Edge/access switches
    info('*** ||||||||| Adding Access Layer....\n')
    edge1 = net.addSwitch('s6')
    edge2 = net.addSwitch('s7')
    edge3 = net.addSwitch('s8')
```

```
    edge4 = net.addSwitch('s9')
    edge5 = net.addSwitch('s10')
    edge6 = net.addSwitch('s11')

    info('*** Adding hosts\n')
    # Hosts for different VLANs
    hosts = []
    vlans = [10, 20, 30, 40]  # VLANs: mgmt, srv, storage, bkp
    for i in range(12):
        vlan_id = vlans[i // 4]  # Assign VLAN based on host number
        ip = '10.{}.{}.{}'.format(vlan_id, i % 4, i % 256)
        host = net.addHost('h%s' % i, ip=ip)
        hosts.append(host)

    info('*** Creating links\n')
    # Core to Aggregation
    net.addLink(core1, agg1)
    net.addLink(core1, agg2)
    net.addLink(core2, agg2)
    net.addLink(core2, agg3)

    # Aggregation to Edge
    net.addLink(agg1, edge1)
    net.addLink(agg1, edge2)
    net.addLink(agg2, edge3)
    net.addLink(agg2, edge4)
    net.addLink(agg3, edge5)
    net.addLink(agg3, edge6)

    # Edge to Hosts
    for i in range(2):
        net.addLink(edge1, hosts[i], params1={'vlan': 10})
        net.addLink(edge2, hosts[i+2], params1={'vlan': 20})
        net.addLink(edge3, hosts[i+4], params1={'vlan': 30})
        net.addLink(edge4, hosts[i+6], params1={'vlan': 40})
        net.addLink(edge5, hosts[i+8], params1={'vlan': 30})
        net.addLink(edge6, hosts[i+10], params1={'vlan': 40})

    info('*** Starting network\n')
    net.start()

    info('*** Configuring VLANs on switches\n')
    # Configure VLANs on edge switches
    edge_switches = [edge1, edge2, edge3, edge4, edge5, edge6]
    for i, edge in enumerate(edge_switches):
        vlan_id = vlans[i // 2]
        edge.cmd('ovs-vsctl add-port %s %s-tagged -- set port %s-tagged tag=%d' %
(edge.name, edge.name, edge.name, vlan_id))

    info('*** Running CLI\n')
    CLI(net)

    info('*** Stopping network\n')
    net.stop()
```

```python
if __name__ == '__main__':
    setLogLevel('info')
    createDataCenter()
```

Running the script:

```
Activities    Terminal                         Jun 20  02:57
kalindu@kalindu-Inspiron-13-5310: ~/SEM-6/CO
kalindu@kalindu-Inspiron-13-5310: ~/SEM-6/CO515/Lab9
kalindu@kalindu-Inspiron-13-5310:~/SEM-6/CO515/Lab9$ sudo vim datacenter-sdn.py
kalindu@kalindu-Inspiron-13-5310:~/SEM-6/CO515/Lab9$ sudo python3 datacenter-sdn.py
*** Adding controller
*** Adding switches
*** ||||||||| Adding Core Layer....
*** ||||||||| Adding Aggregation Layer....
*** ||||||||| Adding Access Layer....
*** Adding hosts
*** Creating links
*** Starting network
*** Configuring hosts
h0 h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11
*** Starting controller
c0
*** Starting 11 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 ...
*** Configuring VLANs on switches
*** Running CLI
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h0 h1 h10 h11 h2 h3 h4 h5 h6 h7 h8 h9 s1 s10 s11 s2 s3 s4 s5 s6 s7 s8 s9
```

Figure 33: Starting the Mininet DCN

The following output indicates that all the intended connections for our 3-tier architecture has been properly configured.

```
mininet> net
h0 h0-eth0:s6-eth2
h1 h1-eth0:s6-eth3
h2 h2-eth0:s7-eth2
h3 h3-eth0:s7-eth3
h4 h4-eth0:s8-eth2
h5 h5-eth0:s8-eth3
h6 h6-eth0:s9-eth2
h7 h7-eth0:s9-eth3
h8 h8-eth0:s10-eth2
h9 h9-eth0:s10-eth3
h10 h10-eth0:s11-eth2
h11 h11-eth0:s11-eth3
s1 lo:  s1-eth1:s3-eth1 s1-eth2:s4-eth1
s2 lo:  s2-eth1:s4-eth2 s2-eth2:s5-eth1
s3 lo:  s3-eth1:s1-eth1 s3-eth2:s6-eth1 s3-eth3:s7-eth1
s4 lo:  s4-eth1:s1-eth2 s4-eth2:s2-eth1 s4-eth3:s8-eth1 s4-eth4:s9-eth1
s5 lo:  s5-eth1:s2-eth2 s5-eth2:s10-eth1 s5-eth3:s11-eth1
s6 lo:  s6-eth1:s3-eth2 s6-eth2:h0-eth0 s6-eth3:h1-eth0
s7 lo:  s7-eth1:s3-eth3 s7-eth2:h2-eth0 s7-eth3:h3-eth0
s8 lo:  s8-eth1:s4-eth3 s8-eth2:h4-eth0 s8-eth3:h5-eth0
s9 lo:  s9-eth1:s4-eth4 s9-eth2:h6-eth0 s9-eth3:h7-eth0
s10 lo:  s10-eth1:s5-eth2 s10-eth2:h8-eth0 s10-eth3:h9-eth0
s11 lo:  s11-eth1:s5-eth3 s11-eth2:h10-eth0 s11-eth3:h11-eth0
c0
mininet>
```

Figure 34: Connections in the implemented three tier architecture

Cleaning the Mininet Network setup:

```
mininet>
mininet> exit
*** Stopping network
*** Stopping 1 controllers
c0
*** Stopping 22 links
......................
*** Stopping 11 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11
*** Stopping 12 hosts
h0 h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11
*** Done
kalindu@kalindu-Inspiron-13-5310:~/SEM-6/CO515/Lab9$ S
```

2. Summary of network virtualization benefits(benefits of SDN in data center environments)

- ***Centralized Management:***
    - *SDN allows for centralized control of the network through a single controller, simplifying network management and enabling more efficient use of resources. Network administrators can manage the entire network infrastructure from a single point of control, which reduces complexity and enhances visibility.*

- ***Flexibility and Scalability:***
    - *SDN enables dynamic provisioning and management of network resources, allowing data centers to scale quickly to meet changing demands. Network resources can be allocated or reallocated on-demand, enabling the data center to respond rapidly to new workloads or changing application requirements.*

- ***Improved Network Efficiency:***
    - *With SDN, traffic can be routed optimally, reducing congestion and improving overall network performance. SDN also allows for automated network configuration and management, reducing human error and downtime. By centralizing control, SDN ensures that data flows are optimized across the network, enhancing throughput and reducing latency.*

- ***Enhanced Security:***
    - *SDN provides improved security through centralized security policies and real-time threat detection. Security policies can be uniformly enforced across the entire network from the central controller. Additionally, SDN can dynamically adapt to security threats by rerouting traffic or isolating compromised sections of the network, providing a robust defense against attacks.*

- ***Cost Efficiency:***
    - *SDN can reduce operational and capital expenditures by utilizing commodity hardware and automating routine tasks. The decoupling of the control and data planes allows for the use of less expensive hardware while maintaining high performance. Automation reduces the need for manual intervention, which lowers the risk of errors and the associated costs of troubleshooting and downtime.*