# MULTIPATH LOAD BALANCING

## FOR SDN

ADVANCED COMPUTER NETWORKS

AUTHOR
DHARSHAN KUMAR K S

# IMPLEMENTATIONS DONE

01

LOAD BALANCING

02

PATH FINDING

03

INSTALLING FLOW

04

SDN DEMONSTRATION

# 01

# LOAD BALANCING

## Types of Load Balancing

- Round Robin
- Random
- Multipath
- Least Connections
- Weighted Round Robin

# Multipath Load Balancing

Method of managing incoming traffic by distributing and sharing load fairly among multiple routes from source to destination hosts

# Load Balancing in SDN

**Network Virtualization:**
Instead of middlebox/hardware, we can implement in software through program codes

SDN controller have built-in load balancers

**Goal:**
- Reduce network response time
- Network Utility & Make use of idle hosts
- Harder to sniff packets gives Network security
- Increases Bandwidth due to parallel transfer

# Load Balancing in SDN

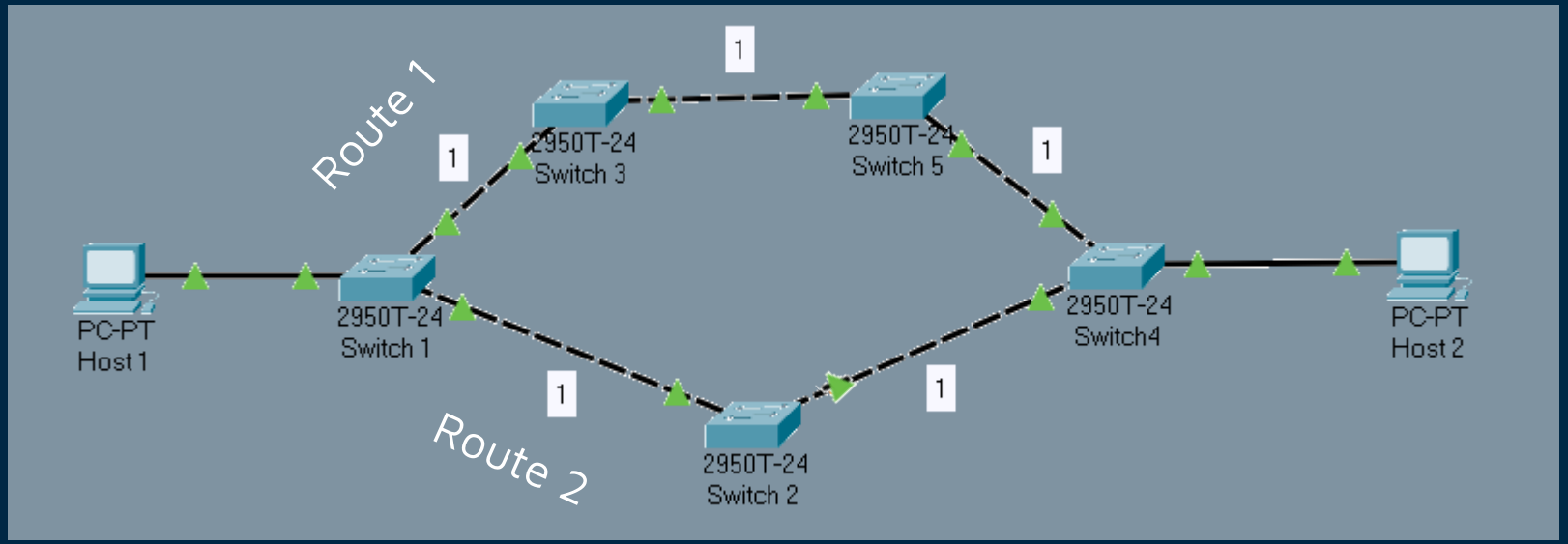
OpenFlow protocol v1.1 support Group Tables
Apply multiple actions to a specific flow

All – Multicast
Select – Load sharing
Indirect – Indirection
Fast Failover – Rerouting

# 02

# PATH FINDING

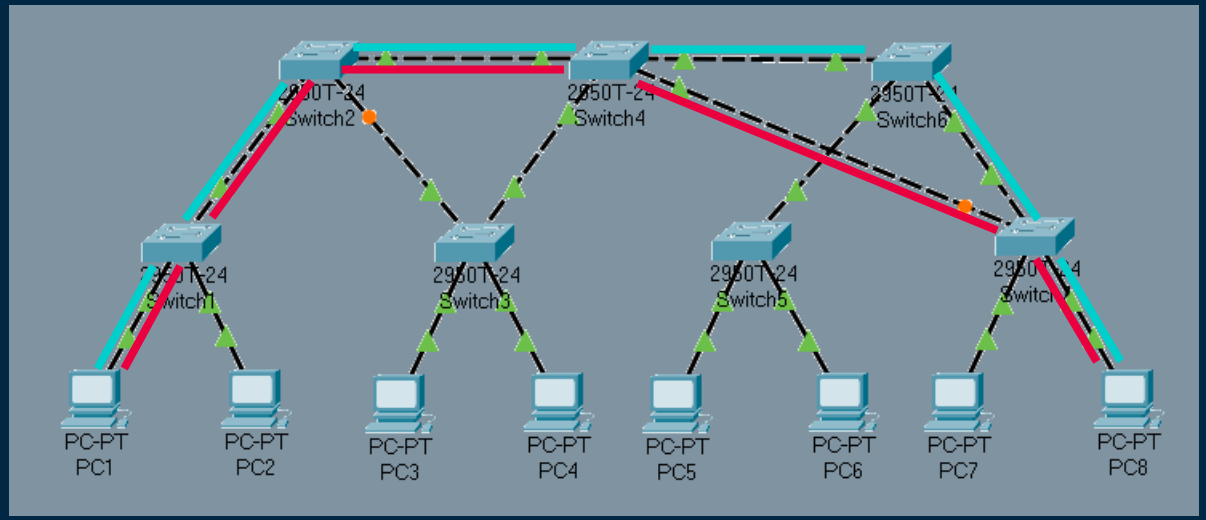
# Path Finding




Source- host1
Destination- host2

Path1 – {S1-S2-S4}
Path2 – {S1-S3-S5-S4}

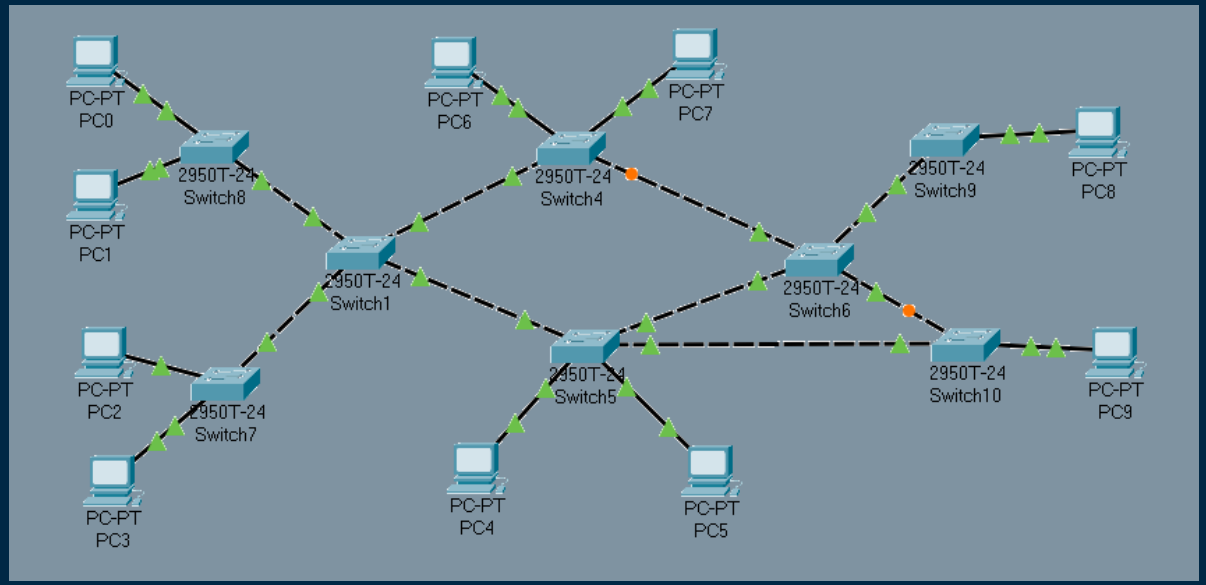
# Topology-2




Source- host1
Destination- host8

Path1 – {S1-S2-S4-S7}
Path2 – {S1-S2-S4-S6-S7}
Path3 – {S1-S2-S3-S4-S7}
Path4 – {S1-S2-S3-S4-S6-S7}

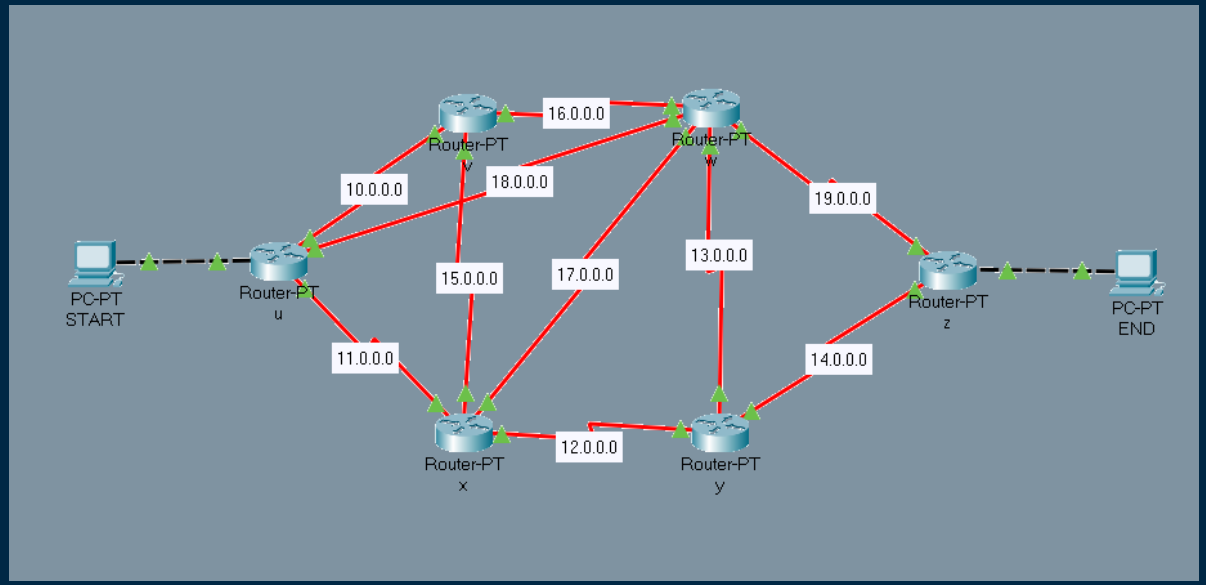
# Topology-3




Source- host1
Destination- host8

Path1 – {S8-S1-S4-S6-S9}
Path2 – {S8-S1-S5-S6-S9}
Path3 – {S8-S1-S5-S10-S6-S9}

# Topology-4



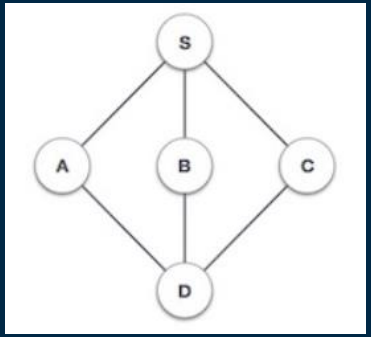

Source- host1
Destination- host2

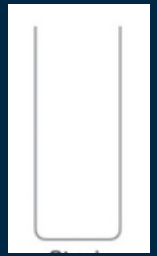
# Path Finding using DFS

1) Traverse depth-wise & Store all switches of a route in a stack
2) Find Deepest switch in network
3) Backtracks to find initial switch & find other deepest switch
4) List all routes

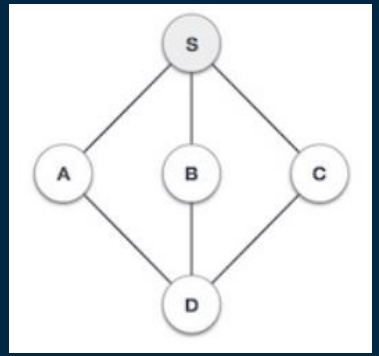
Time Complexity: O(V+E)

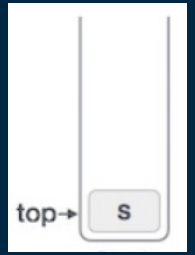
Visit adjacent unvisited vertex. Make it as visited & Insert it into stack
If no adjacent vertex, remove last vertex from stack



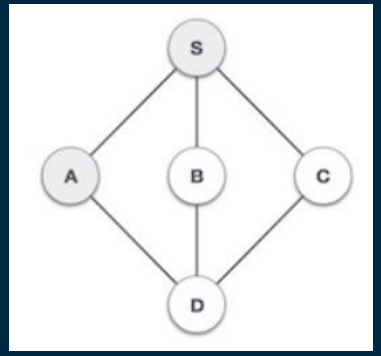



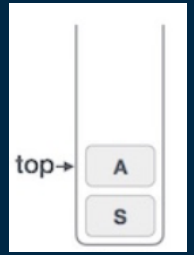
Stack



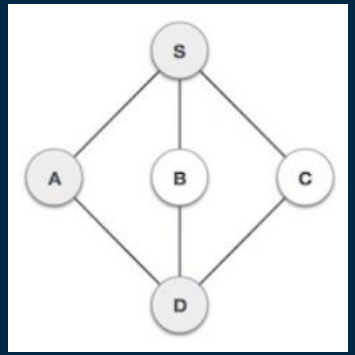



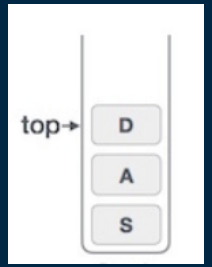

Stack



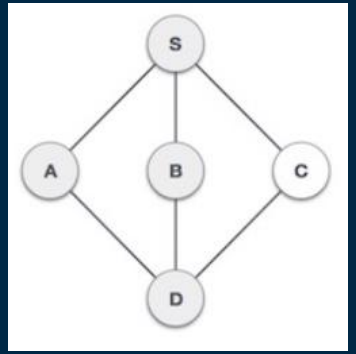




Stack



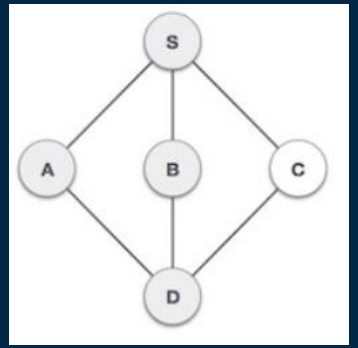



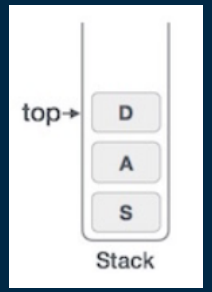

Stack



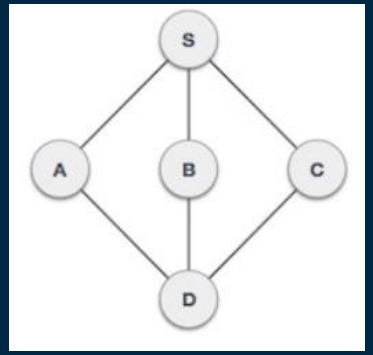


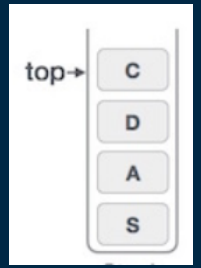
Stack



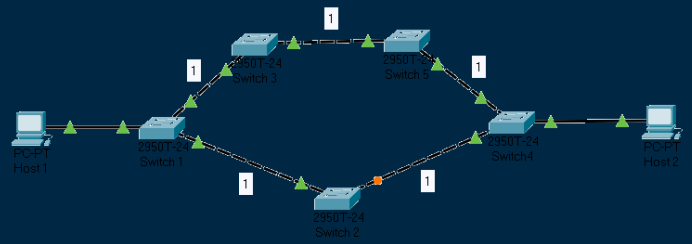

Stack




Stack

# Path Finding using DFS




```python
def get_paths(self, src, dst):
    '''
    Get all paths from src to dst using DFS algorithm
    '''
    if src == dst:
        # host target is on the same switch
        return [[src]]
    paths = []
    stack = [(src, [src])]
    while stack:
        (node, path) = stack.pop()
        for next in set(self.adjacency[node].keys()) - set(path):
            if next is dst:
                paths.append(path + [next])
            else:
                stack.append((next, path + [next]))
    print "Available paths from ", src, " to ", dst, " : ", paths
    return paths
```

# 02

# PATH COST FINDING

DFS returns only path

But to calculate Bucket weights, we need path cost

## Link Cost

```
def get_link_cost(self, s1, s2):
    '''
    Get the link cost between two switches
    '''
    e1 = self.adjacency[s1][s2]
    e2 = self.adjacency[s2][s1]
    bl = min(self.bandwidths[s1][e1], self.bandwidths[s2][e2])
    ew = REFERENCE_BW/bl
    return ew
```

## Path Cost

```
def get_path_cost(self, path):
    '''
    Get the path cost
    '''
    cost = 0
    for i in range(len(path) - 1):
        cost += self.get_link_cost(path[i], path[i+1])
    return cost
```

# Optimal Path

```
def get_optimal_paths(self, src, dst):
    '''
    Get the n-most optimal paths according to MAX_PATHS
    '''
    paths = self.get_paths(src, dst)
    paths_count = len(paths) if len(
        paths) &amp;amp;amp;amp;amp;amp;amp;gt; MAX_PATHS else MAX_PATHS
    return sorted(paths, key=lambda x: self.get_path_cost(x))[0:(paths_count)]
```

# 03

# INSTALLING FLOW IN SWITCH

# Flow Entry

A set of actions to be applied based on a criteria of the packet headers

Match & Action uses the 5 packet headers and the flow entry in flow table to forward packet across multiple ports

| Port | MAC src | MAC dest | Eth Type | …… | Src IP | Dest IP | …….. | Action |
|---|---|---|---|---|---|---|---|---|
| 1 | 00:00:5e:00:53:af | 00:00:5e:00:64:bg | 0x0800 | | 192.13.4.2 | 192.14.1.1 | | Group 100 |

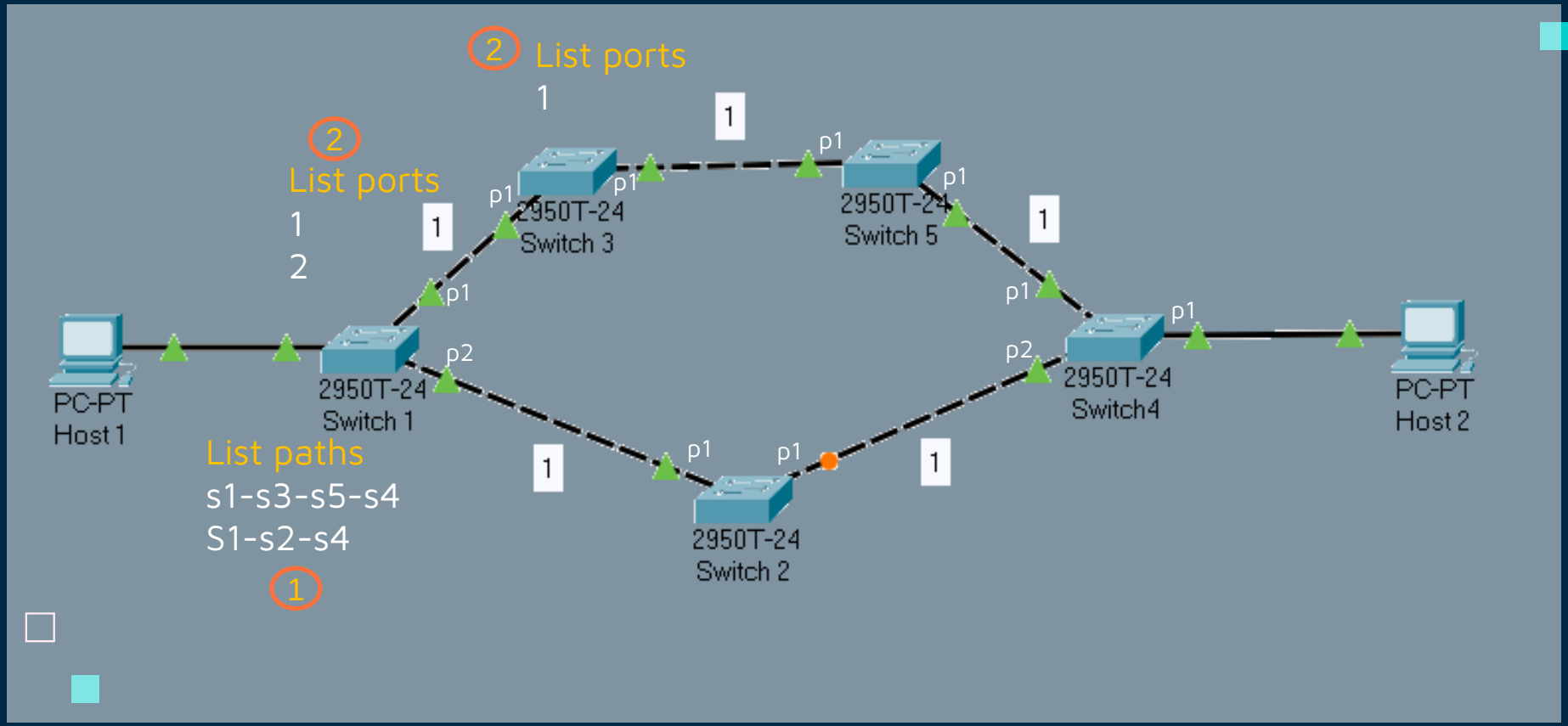
# Workflow

1) List all paths from source to destination.
2) Loop through all the switches that contain a path.
    1) List all the ports in the switch that contains a path.
    2) If (multiple ports in switch contain a path){
        Create a group table flow with type select }
    else {
        Install a normal flow }

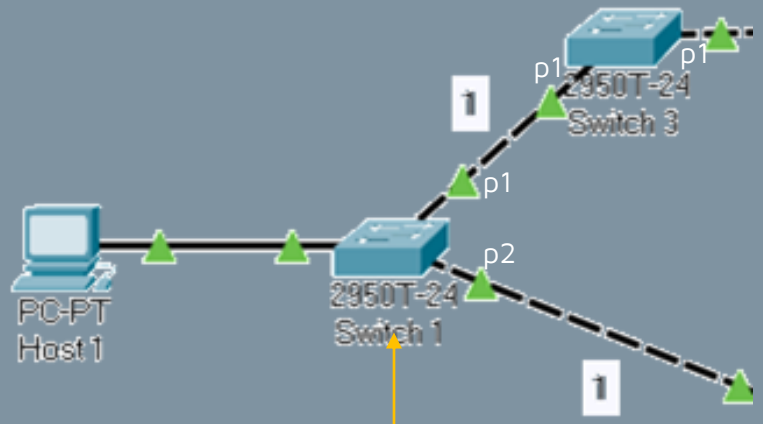
To create a group table, we create buckets (group of actions)
bucket weight: weight of the bucket
actions: output ports

**install_path(self, src, first_port, dst, last_port, ip_src, ip_dst)**

# Workflow


2
List ports
1
2
List ports
1
2
1
1
1
1
p1
p1
p1
p1
p1
p1
p1
p2
p2
p1
p1
2950T-24
Switch 3
2950T-24
Switch 5
2950T-24
Switch 1
2950T-24
Switch 2
2950T-24
Switch4
PC-PT
Host 1
PC-PT
Host 2
List paths
s1-s3-s5-s4
S1-s2-s4
1

# Workflow

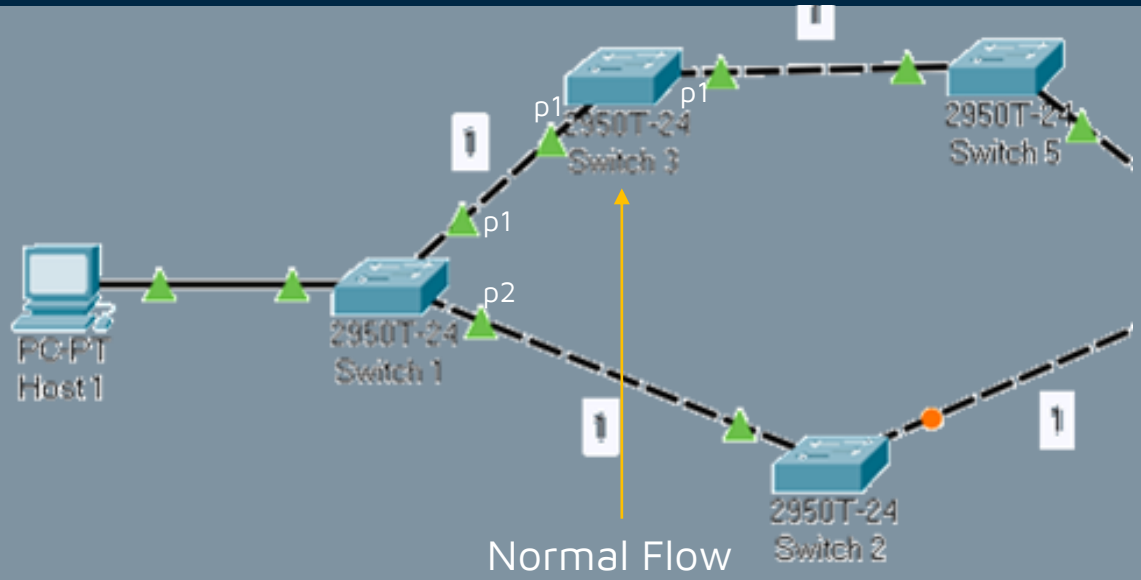



Group Table

| Port | MAC src | …… … | Dst IP | …….. . | Action |
|---|---|---|---|---|---|
| 1 | * | | 192.14.1.1 | | Group 100 |

| Group ID | Group Type | Counter | Action Buckets |
|---|---|---|---|
| 100 | Select | 1 | Port1, Port2 |

# Workflow




| Port | MAC src | …… … | Dst IP | …….. . | Action |
|---|---|---|---|---|---|
| 1 | * | | 192.14.1.1 | | Port1 |

# Bucket weight

ratio of the path weight of p with the total path weight of the available paths

$$bw(p) = \left(1 - \frac{pw(p)}{\sum_{i<n}^{i=0} pw(i)}\right) x10$$

For a path p :

bw - bucket weight, 0 ≤ bw(p) < 10

pw - path weight/cost

n - total no. of paths available

# Bucket weight for our topology

**Path Weight:**

pw1 = (s4-s2) + (s2-s1) = 2

pw2 = (s4-s5) + (s5+s3) + (s3-s1) = 3

**Bucket Weight:**

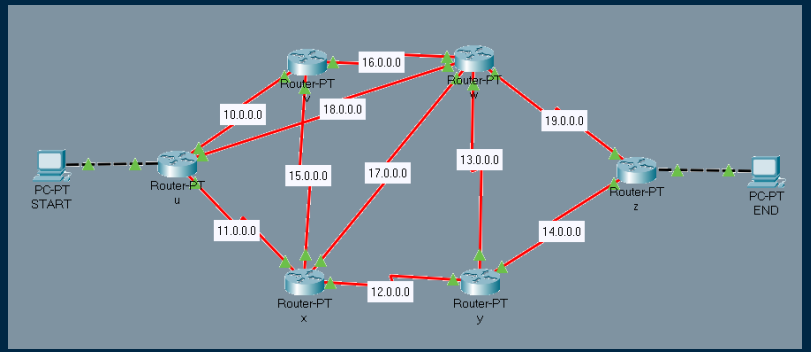
bw1 = (1 – 2/5) * 10 = 6

bw2 = (1 – 3/5) * 10 = 4

# 04

# SDN DEMONSTRATION

# Mininet Topology




```python
from mininet.topo import Topo

class MyTopo( Topo ):
    "ring topology example."
    def build( self ):
        "Create custom topo."

        # Add hosts
        h1 = self.addHost('h1')
        h2 = self.addHost('h2')

        #Add switches
        u = self.addSwitch('s1')
        v = self.addSwitch('s2')
        x = self.addSwitch('s3')
        y = self.addSwitch('s4')
        w = self.addSwitch('s5')
        z = self.addSwitch('s6')

        # Add links
        self.addLink( h1, u )
        self.addLink( u, v )
        self.addLink( u, x )
        self.addLink( u, w )
        self.addLink( v, w )
        self.addLink( v, x )
        self.addLink( x, w )
        self.addLink( x, y )
        self.addLink( y, w )
        self.addLink( y, z )
        self.addLink( z, w )
        self.addLink( z, h2 )

topos = { 'dynamic_topo': ( lambda: MyTopo() ) }
```

# RYU controller

dharshan_kumar@dharshan: ~/ryu/Ryu-Multipath

dharshan_kumar@dharshan: ~ | dharshan_kumar@dharshan: ~/ryu/Ryu-Multipath

```
dharshan_kumar@dharshan:~/ryu/Ryu-Multipath$ ryu-manager --observe-links ryu_multipath.py
loading app ryu_multipath.py
loading app ryu.topology.switches
loading app ryu.controller.ofp_handler
instantiating app ryu_multipath.py of ProjectController
instantiating app ryu.topology.switches of Switches
instantiating app ryu.controller.ofp_handler of OFPHandler
switch_features_handler is called
switch_features_handler is called
switch_features_handler is called
switch_features_handler is called
switch_features_handler is called
switch_features_handler is called
```

# Ping

dharshan_kumar@dharshan: ~/mininet/custom

dharshan_kumar@dharshan: ~/mininet/custom | dharshan_kumar@dharshan: ~/ryu/Ryu-Multipath

```
dharshan_kumar@dharshan:~/mininet/custom$ sudo mn --custom ex_dynamicrouting.py --topo dynamic_topo
--controller=remote
*** Creating network
*** Adding controller
Connecting to remote controller at 127.0.0.1:6653
*** Adding hosts:
h1 h2
*** Adding switches:
s1 s2 s3 s4 s5 s6
*** Adding links:
(h1, s1) (s1, s2) (s1, s3) (s1, s5) (s2, s3) (s2, s5) (s3, s4) (s3, s5) (s4, s5) (s4, s6) (s6, h2) (
s6, s5)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 6 switches
s1 s2 s3 s4 s5 s6 ...
*** Starting CLI:
mininet> h1 ping h2 -c4
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=135 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.258 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.263 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.116 ms

--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3051ms
rtt min/avg/max/mdev = 0.116/33.915/135.023/58.374 ms
mininet>
```

# Path Finding

dharshan_kumar@dharshan: ~/ryu/Ryu-Multipath

dharshan_kumar@dharshan: ~/mininet/custom | dharshan_kumar@dharshan: ~/ryu/Ryu-Multipath

```
Available paths from  6  to  1  :  [[6, 5, 1], [6, 5, 4, 3, 1], [6, 5, 4, 3, 2, 1], [6, 5, 3, 1], [6
, 5, 3, 2, 1], [6, 5, 2, 1], [6, 5, 2, 3, 1], [6, 4, 5, 1], [6, 4, 5, 3, 1], [6, 4, 5, 3, 2, 1], [6,
 4, 5, 2, 1], [6, 4, 5, 2, 3, 1], [6, 4, 3, 1], [6, 4, 3, 5, 1], [6, 4, 3, 5, 2, 1], [6, 4, 3, 2, 1]
, [6, 4, 3, 2, 5, 1]]
[6, 5, 1] cost =  2.0
[6, 5, 3, 1] cost =  3.0
[(1, 2.0)]
[(1, 3.0)]
[(1, 3.0)]
[(1, 2.0), (3, 3.0)]
[(2, 2.0), (2, 3.0)]
Path installation finished in  0.005845546722412109
Available paths from  1  to  6  :  [[1, 5, 6], [1, 5, 4, 6], [1, 5, 3, 4, 6], [1, 5, 2, 3, 4, 6], [1
, 3, 5, 6], [1, 3, 5, 4, 6], [1, 3, 4, 6], [1, 3, 4, 5, 6], [1, 3, 2, 5, 6], [1, 3, 2, 5, 4, 6], [1,
 2, 5, 6], [1, 2, 5, 4, 6], [1, 2, 5, 3, 4, 6], [1, 2, 3, 5, 6], [1, 2, 3, 5, 4, 6], [1, 2, 3, 4, 6]
, [1, 2, 3, 4, 5, 6]]
[1, 5, 6] cost =  2.0
[1, 5, 4, 6] cost =  3.0
[(4, 2.0), (4, 3.0)]
[(3, 3.0)]
[(5, 2.0), (4, 3.0)]
[(3, 2.0)]
[(3, 3.0)]
Path installation finished in  0.004446268081665039
```

# Flow Entry

```
mininet> sh ovs-ofctl dump-flows s1
 cookie=0x0, duration=1258.609s, table=0, n_packets=2980, n_bytes=178800, priority=65535,dl_dst=01:8
0:c2:00:00:0e,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x0, duration=1255.317s, table=0, n_packets=4, n_bytes=392, ip,nw_src=10.0.0.2,nw_dst=10.0.0
.1 actions=output:"s1-eth1"
 cookie=0x0, duration=1255.317s, table=0, n_packets=13, n_bytes=546, priority=1,arp,arp_spa=10.0.0.2
,arp_tpa=10.0.0.1 actions=output:"s1-eth1"
 cookie=0x0, duration=1255.317s, table=0, n_packets=4, n_bytes=392, ip,nw_src=10.0.0.1,nw_dst=10.0.0
.2 actions=group:1488893676
 cookie=0x0, duration=1255.317s, table=0, n_packets=2, n_bytes=84, priority=1,arp,arp_spa=10.0.0.1,a
rp_tpa=10.0.0.2 actions=group:1488893676
 cookie=0x0, duration=1258.544s, table=0, n_packets=105, n_bytes=11467, priority=1,ipv6 actions=drop
 cookie=0x0, duration=1258.639s, table=0, n_packets=12, n_bytes=614, priority=0 actions=CONTROLLER:6
5535
mininet>
```

# Group Table Entry

```
mininet> sh ovs-ofctl dump-groups s1
NXST_GROUP_DESC reply (xid=0x2):
 group_id=1488893676,type=select,bucket=bucket_id:0,weight:6,watch_port:"s1-eth4",actions=output:"s1
-eth2",bucket=bucket_id:1,weight:4,watch_port:"s1-eth4",actions=output:"s1-eth3"
mininet>
```

# Simulate Server & Client

"Node: h1"

```
root@dharshan:/home/dharshan_kumar/mininet/custom# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 25] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 43756
[ 24] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 43754
[ 26] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 43758
[ 27] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 43760
[ ID] Interval       Transfer     Bandwidth
[ 27]  0.0-10.0 sec  10.0 GBytes  8.61 Gbits/sec
[ 24]  0.0-10.0 sec  10.6 GBytes  9.09 Gbits/sec
[ 25]  0.0-10.1 sec  11.3 GBytes  9.64 Gbits/sec
[ 26]  0.0-10.1 sec  9.35 GBytes  7.99 Gbits/sec
[SUM]  0.0-10.1 sec  41.3 GBytes  35.3 Gbits/sec
```

"Node: h2"

```
root@dharshan:/home/dharshan_kumar/mininet/custom# iperf -c 10.0.0.1 -P 4
------------------------------------------------------------
Client connecting to 10.0.0.1, TCP port 5001
TCP window size: 2.28 MByte (default)
------------------------------------------------------------
[ 24] local 10.0.0.2 port 43756 connected with 10.0.0.1 port 5001
[ 26] local 10.0.0.2 port 43760 connected with 10.0.0.1 port 5001
[ 23] local 10.0.0.2 port 43754 connected with 10.0.0.1 port 5001
[ 25] local 10.0.0.2 port 43758 connected with 10.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 26]  0.0-10.0 sec  10.0 GBytes  8.62 Gbits/sec
[ 23]  0.0-10.0 sec  10.6 GBytes  9.13 Gbits/sec
[ 24]  0.0-10.0 sec  11.3 GBytes  9.67 Gbits/sec
[ 25]  0.0-10.0 sec  9.35 GBytes  8.02 Gbits/sec
[SUM]  0.0-10.0 sec  41.3 GBytes  35.4 Gbits/sec
root@dharshan:/home/dharshan_kumar/mininet/custom#
```

# Simulate Server & Client

```
mininet> sh ovs-ofctl dump-ports s1
OFPST_PORT reply (xid=0x2): 5 ports
  port LOCAL: rx pkts=0, bytes=0, drop=3, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
  port  "s1-eth4": rx pkts=402, bytes=25673, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=401, bytes=25631, drop=0, errs=0, coll=0
  port  "s1-eth1": rx pkts=213568, bytes=14098664, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=865668, bytes=48796831085, drop=0, errs=0, coll=0
  port  "s1-eth2": rx pkts=419792, bytes=23412140381, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=162591, bytes=10732711, drop=0, errs=0, coll=0
  port  "s1-eth3": rx pkts=446275, bytes=25384716251, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=51769, bytes=3416363, drop=0, errs=0, coll=0
mininet>
```

Port1 (tx): 865668
Port2 (rx): 419792
Port3 (rx): 446275

Obtained Ratio
419792 : 446275 = 2 : 1.7 = 6 : 5.1

Original Ratio
6:4

6 : 5.1 ~ 6 : 4

# THANK YOU

Contact Dharshan Kumar for any further queries