

SDN Lab Guide: Dynamic Traffic Throttling & REST API Verification

Q1. Dynamic Traffic Throttling using Mininet and Ryu

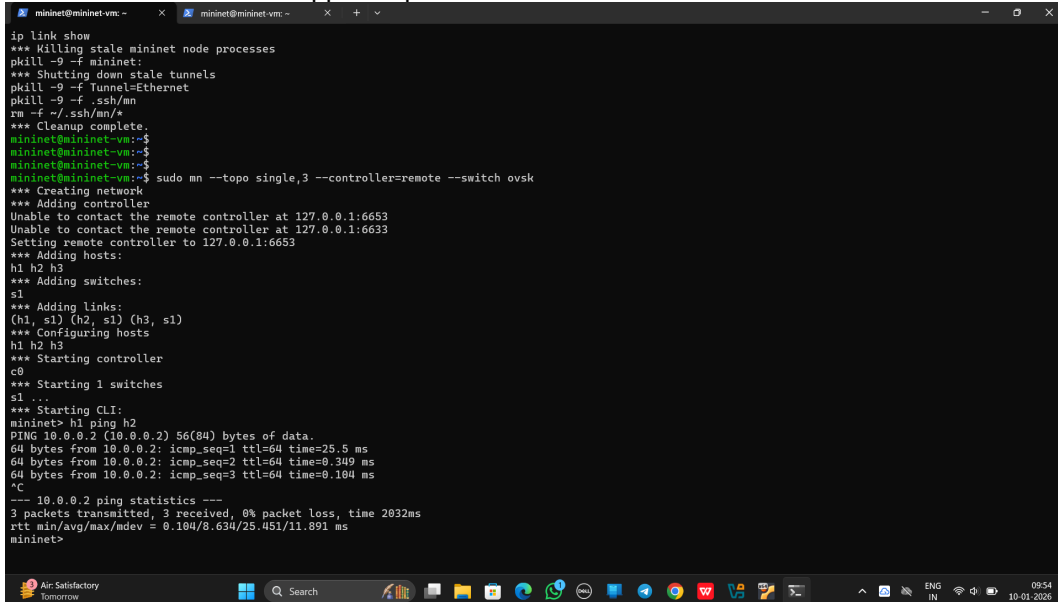
This experiment demonstrates how Software Defined Networking (SDN) enables dynamic traffic throttling using OpenFlow meters. A simple Mininet topology is controlled by a Ryu controller. Traffic from a selected host is rate-limited during peak usage without restarting the network.

Topology Description

The topology consists of one OpenFlow-enabled switch (s1) and three hosts (h1, h2, h3). Host h1 generates traffic towards h2. The switch enforces bandwidth limits using OpenFlow meters.

Normal Traffic Behavior

Initially, no traffic restrictions are applied. Iperf tests show maximum achievable bandwidth between hosts.



```
mininet@mininet-vm: ~
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
mininet@mininet-vm: ~$
mininet@mininet-vm: ~$
mininet@mininet-vm: ~$
mininet@mininet-vm: ~$ sudo mn --topo single,3 --controller=remote --switch ovsk
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6653
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> h1 ping h2
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=25.5 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.349 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.104 ms
^C
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2032ms
rtt min/avg/max/mdev = 0.104/8.634/25.451/11.891 ms
mininet>
```

Applying Meter and Flow Rules

An OpenFlow meter is configured on switch s1 to limit traffic to 1 Mbps. A high-priority flow rule attaches this meter to traffic sourced from host h1.

```

mininet@mininet-vm: ~$
[ 3] 4.0- 5.0 sec 127 KBytes 1.04 Mb/s/sec
[ 3] 5.0- 6.0 sec 127 KBytes 1.04 Mb/s/sec
[ 3] 6.0- 7.0 sec 127 KBytes 1.04 Mb/s/sec
[ 3] 7.0- 8.0 sec 191 KBytes 1.56 Mb/s/sec
[ 3] 8.0- 9.0 sec 63.6 KBytes 521 Kb/s/sec
[ 3] 9.0-10.0 sec 127 KBytes 1.04 Mb/s/sec
[ 3] 10.0-11.0 sec 63.6 KBytes 521 Kb/s/sec
[ 3] 11.0-12.0 sec 127 KBytes 1.04 Mb/s/sec
[ 3] 12.0-13.0 sec 127 KBytes 1.04 Mb/s/sec
[ 3] 13.0-14.0 sec 127 KBytes 1.04 Mb/s/sec
[ 3] 14.0-15.0 sec 127 KBytes 1.04 Mb/s/sec
[ 3] 15.0-16.0 sec 127 KBytes 1.04 Mb/s/sec
[ 3] 16.0-17.0 sec 63.6 KBytes 521 Kb/s/sec
[ 3] 17.0-18.0 sec 191 KBytes 1.56 Mb/s/sec
[ 3] 18.0-19.0 sec 63.6 KBytes 521 Kb/s/sec
[ 3] 19.0-20.0 sec 127 KBytes 1.04 Mb/s/sec
[ 3] 0.0-20.4 sec 2.67 MBytes 1.10 Mb/s/sec
mininet> h2 pkill iperf
-----
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
-----
[ 0] local 10.0.0.2 port 5001 connected with 10.0.0.1 port 49500
[ 10] Interval Transfer Bandwidth
[ 0] 0.0-21.5 sec 2.67 MBytes 1.04 Mb/s/sec
mininet> sh ovs-ofctl -O OpenFlow13 dump-meters s1
OFPST_METER_CONFIG reply (OF1.3) (xid=0x2):
meter=1 kbps band=
type=drop rate=1000
mininet> sh ovs-ofctl -O OpenFlow13 dump-flows s1
cookie=0x0, duration=110.275s, table=0, n_packets=2094, n_bytes=3648972, priority=100,ip,nw_src=10.0.0.1 actions=meter:1,NORMAL
cookie=0x0, duration=175.246s, table=0, n_packets=1706, n_bytes=115660, priority=1,in_port="s1-eth2",dl_src=ca:bf:49:f5:d2:d5,dl_dst=aa:9b:3f:89:56:10 actions=
onsoutput:"s1-eth1"
cookie=0x0, duration=175.237s, table=0, n_packets=4, n_bytes=280, priority=1,in_port="s1-eth1",dl_src=aa:9b:3f:89:56:10,dl_dst=ca:bf:49:f5:d2:d5 actions=ou
tput:"s1-eth2"
cookie=0x0, duration=182.895s, table=0, n_packets=3, n_bytes=182, priority=0 actions=CONTROLLER:65535
mininet>

```

Throttled Traffic Observation

After applying the meter, iperf results clearly show bandwidth limited to approximately 1 Mbps, confirming successful traffic throttling.

```
mininet@mininet-vmc ~$ mininet@mimnet-vmc - x + v
```

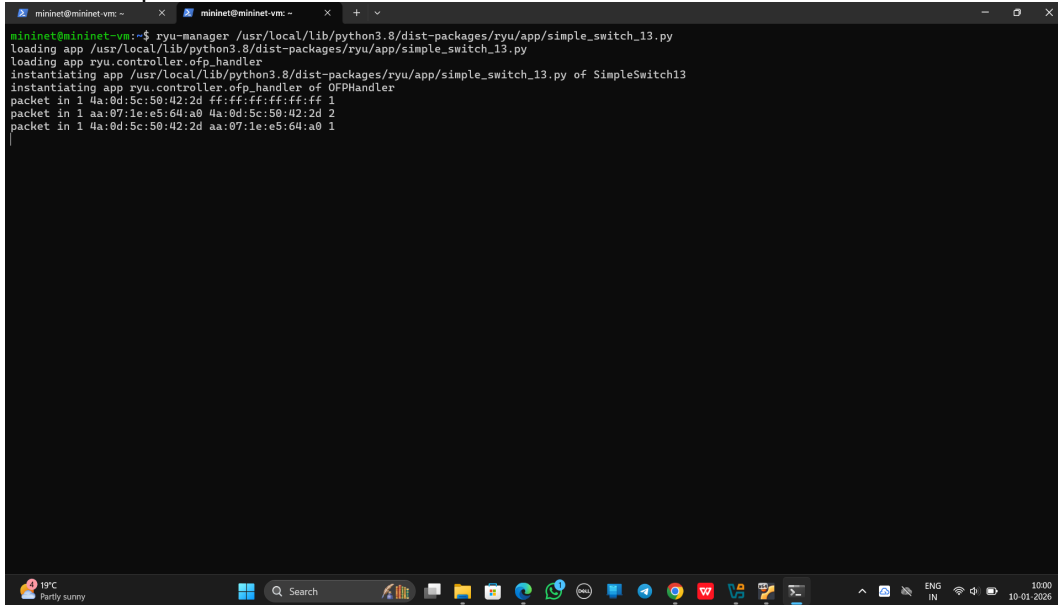
```
mininet> sh ovs-ofctl -O OpenFlow13 dump-meters s1  
OFPSI_METER_CONFIG reply (OF1.3) {xid=0x2}:  
meter=1 kbps band=  
type=drop rate=1000  
  
mininet> sh ovs-ofctl -O OpenFlow13 add-flow s1 priority=100,ip,nw_src=10.0.0.1,actions=meter:1:normal  
mininet> xterm h1  
Error: Cannot connect to display  
mininet> xterm h1 h2  
Error: Cannot connect to display  
Error: Cannot connect to display  
mininet> h2 iperf -s &  
mininet> h1 iperf -c 10.0.0.2 -t 20 -i 1  
-----  
Client connecting to 10.0.0.2, TCP port 5801  
TCP window size: 170 KByte (default)  
-----  
[ ] local 10.0.0.1 port 51698 connected with 10.0.0.2 port 5801  
[ ID] Interval      Transfer    Bandwidth  
[   ] 0.0 - 1.0 sec     511 KBytes  4.18 Mb/sec  
[   ] 1.0 - 2.0 sec     127 KBytes  1.04 Mb/sec  
[   ] 2.0 - 3.0 sec     127 KBytes  1.04 Mb/sec  
[   ] 3.0 - 4.0 sec     63.6 KBytes 521 Kb/sec  
[   ] 4.0 - 5.0 sec     127 KBytes  1.04 Mb/sec  
[   ] 5.0 - 6.0 sec     127 KBytes  1.04 Mb/sec  
[   ] 6.0 - 7.0 sec     127 KBytes  1.04 Mb/sec  
[   ] 7.0 - 8.0 sec     127 KBytes  1.04 Mb/sec  
[   ] 8.0 - 9.0 sec     63.6 KBytes 521 Kb/sec  
[   ] 9.0-10.0 sec     191 KBytes  1.56 Mb/sec  
[   ] 10.0-11.0 sec     63.6 KBytes 521 Kb/sec  
[   ] 11.0-12.0 sec     127 KBytes  1.04 Mb/sec  
[   ] 12.0-13.0 sec     127 KBytes  1.04 Mb/sec  
[   ] 13.0-14.0 sec     127 KBytes  1.04 Mb/sec  
[   ] 14.0-15.0 sec     127 KBytes  1.04 Mb/sec  
[   ] 15.0-16.0 sec     127 KBytes  1.04 Mb/sec  
[   ] 16.0-17.0 sec     127 KBytes  1.04 Mb/sec  
[   ] 17.0-18.0 sec     63.6 KBytes 521 Kb/sec  
[   ] 18.0-19.0 sec     127 KBytes  1.04 Mb/sec  
[   ] 19.0-20.0 sec     127 KBytes  1.04 Mb/sec  
[   ] 0.0-20.4 sec     2.67 MBytes 1.10 Mb/sec  
mininet>
```

Q2. Verifying Controller Decisions using Ryu REST APIs

Ryu provides REST APIs to allow network engineers to inspect the controller's real-time view of the network. These APIs enable validation of controller decisions without requiring topology knowledge beforehand.

Controller Initialization

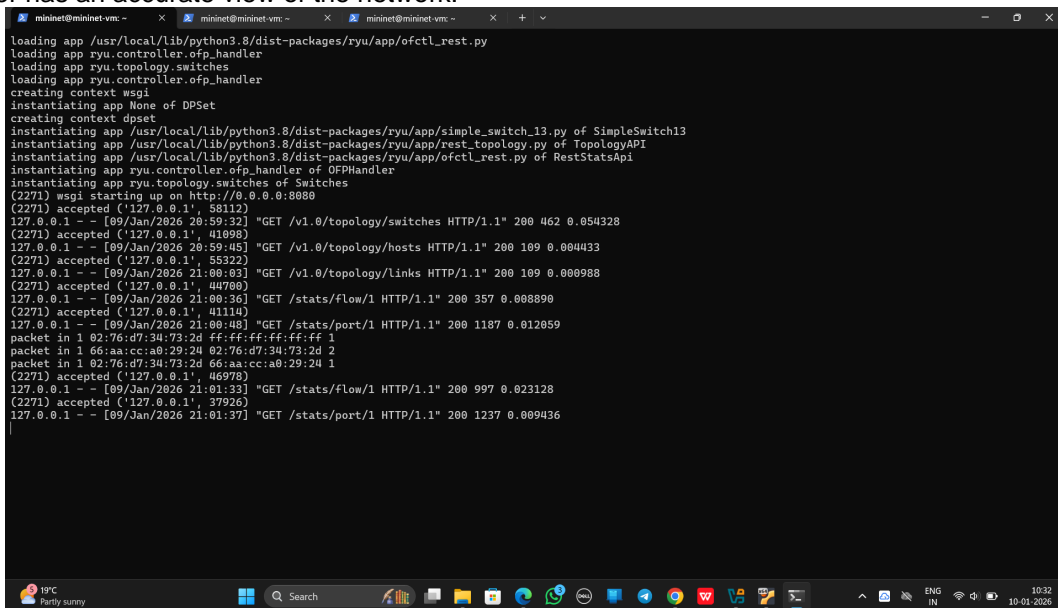
The Ryu controller is launched with REST topology and statistics modules. It listens on port 8080 and responds to HTTP queries.



```
mininet@mininet-vm: ~$ ryu-manager /usr/local/lib/python3.8/dist-packages/ryu/app/simple_switch_13.py
loading app /usr/local/lib/python3.8/dist-packages/ryu/app/simple_switch_13.py
loading app ryu.controller.ofp_handler
instantiating app /usr/local/lib/python3.8/dist-packages/ryu/app/simple_switch_13.py of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
packet in 1 4a:0d:5c:50:42:2d ff:ff:ff:ff:ff:ff 1
packet in 1 aa:07:1e:e5:64:a0 4a:0d:5c:50:42:2d 2
packet in 1 4a:0d:5c:50:42:2d aa:07:1e:e5:64:a0 1
```

Topology Discovery via REST

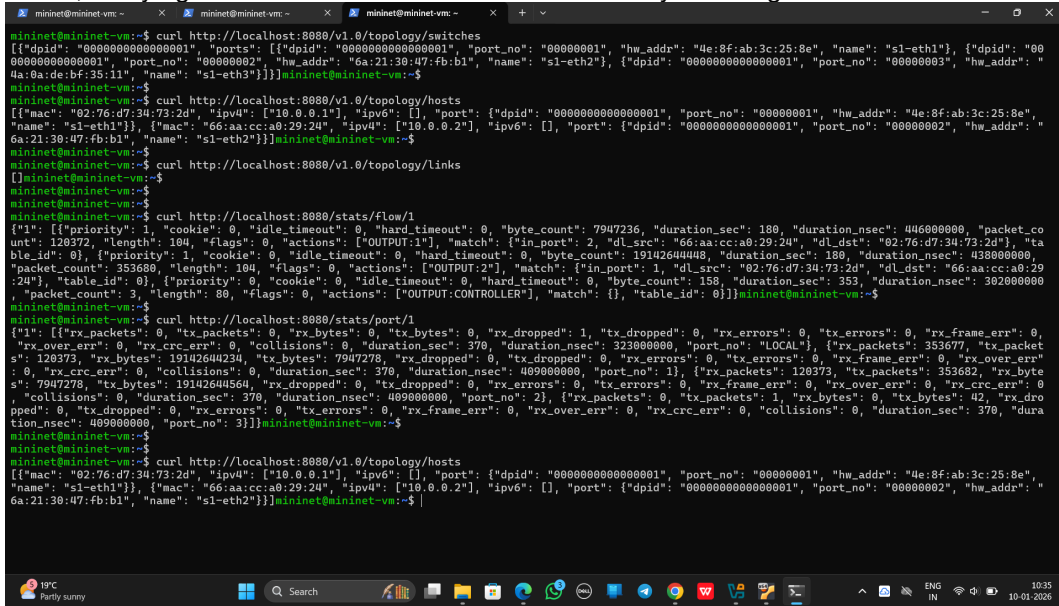
REST endpoints are used to discover switches, hosts, and links. These endpoints confirm that the controller has an accurate view of the network.



```
mininet@mininet-vm: ~$ ryu-manager /usr/local/lib/python3.8/dist-packages/ryu/app/ofctl_rest.py
loading app /usr/local/lib/python3.8/dist-packages/ryu/app/ofctl_rest.py
loading app ryu.controller.ofp_handler
loading app ryu.topology.switches
loading app ryu.controller.ofp_handler
creating context wsgi
instantiating app None of DPSet
instantiating app /usr/local/lib/python3.8/dist-packages/ryu/app/simple_switch_13.py of SimpleSwitch13
instantiating app /usr/local/lib/python3.8/dist-packages/ryu/app/rest_topology.py of TopologyAPI
instantiating app /usr/local/lib/python3.8/dist-packages/ryu/app/ofctl_rest.py of RestStatsApi
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app ryu.topology.switches of Switches
(2271) wsgi starting up on http://0.0.0.0:8080
(2271) accepted ('127.0.0.1', 58112)
127.0.0.1 - - [09/Jun/2026 20:59:32] "GET /v1.0/topology/switches HTTP/1.1" 200 462 0.054328
(2271) accepted ('127.0.0.1', 41098)
127.0.0.1 - - [09/Jun/2026 20:59:45] "GET /v1.0/topology/hosts HTTP/1.1" 200 109 0.004433
(2271) accepted ('127.0.0.1', 55322)
127.0.0.1 - - [09/Jun/2026 21:00:03] "GET /v1.0/topology/links HTTP/1.1" 200 109 0.000988
(2271) accepted ('127.0.0.1', 44780)
127.0.0.1 - - [09/Jun/2026 21:00:36] "GET /stats/flow/1 HTTP/1.1" 200 357 0.008890
(2271) accepted ('127.0.0.1', 41114)
127.0.0.1 - - [09/Jun/2026 21:00:48] "GET /stats/port/1 HTTP/1.1" 200 1187 0.012059
packet in 1 02:76:d7:34:73:2d ff:ff:ff:ff:ff:ff 1
packet in 1 66:aa:cc:a0:29:24 02:76:d7:34:73:2d 2
packet in 1 02:76:d7:34:73:2d 66:aa:cc:a0:29:24 1
(2271) accepted ('127.0.0.1', 46978)
127.0.0.1 - - [09/Jun/2026 21:01:13] "GET /stats/flow/1 HTTP/1.1" 200 997 0.023128
(2271) accepted ('127.0.0.1', 37926)
127.0.0.1 - - [09/Jun/2026 21:01:37] "GET /stats/port/1 HTTP/1.1" 200 1237 0.009436
```

Traffic Statistics Validation

Flow and port statistics retrieved via REST APIs show packet and byte counters increasing during ping and iperf tests, verifying that controller-installed rules are actively handling traffic.



```
mininet@mininet-vm:~$ curl http://localhost:8080/v1.0/topology/switches
[{"dpid": "0000000000000001", "ports": [{"dpid": "0000000000000001", "port_no": "00000001", "hw_addr": "4e:8f:ab:3c:25:8e", "name": "s1-eth1"}, {"dpid": "0000000000000001", "port_no": "00000002", "hw_addr": "6a:21:30:47:fb:b1", "name": "s1-eth2"}, {"dpid": "0000000000000001", "port_no": "00000003", "hw_addr": "6a:21:30:47:fb:b1", "name": "s1-eth3"}]}]mininet@mininet-vm:~$
mininet@mininet-vm:~$ curl http://localhost:8080/v1.0/topology/hosts
[{"mac": "02:76:d7:34:73:2d", "ip4": ["10.0.0.1"], "ip6": [], "port": {"dpid": "0000000000000001", "port_no": "00000001", "hw_addr": "4e:8f:ab:3c:25:8e", "name": "s1-eth1"}, {"mac": "66:aa:cc:a8:29:24", "ip4": ["10.0.0.2"], "ip6": [], "port": {"dpid": "0000000000000001", "port_no": "00000002", "hw_addr": "6a:21:30:47:fb:b1", "name": "s1-eth2"}}]mininet@mininet-vm:~$
mininet@mininet-vm:~$ curl http://localhost:8080/v1.0/topology/links
[]mininet@mininet-vm:~$
mininet@mininet-vm:~$ curl http://localhost:8080/stats/flow/1
{"1": [{"priority": 1, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 7947236, "duration_sec": 180, "duration_nsec": 44600000, "packet_count": 120372, "length": 104, "flags": 0, "actions": [{"OUTPUT:1}], "match": {"in_port": 2, "dl_src": "66:aa:cc:a8:29:24", "dl_dst": "02:76:d7:34:73:2d", "table_id": 0}, {"priority": 1, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 1914264448, "duration_sec": 180, "duration_nsec": 43800000, "packet_count": 353680, "length": 104, "flags": 0, "actions": [{"OUTPUT:2}], "match": {"in_port": 1, "dl_src": "02:76:d7:34:73:2d", "dl_dst": "66:aa:cc:a8:29:24", "table_id": 0}, {"priority": 0, "cookie": 0, "idle_timeout": 0, "hard_timeout": 0, "byte_count": 158, "duration_sec": 353, "duration_nsec": 30200000, "packet_count": 3, "length": 80, "flags": 0, "actions": [{"OUTPUT:CONTROLLER"]}], "match": {}, "table_id": 0}]}mininet@mininet-vm:~$
mininet@mininet-vm:~$ curl http://localhost:8080/stats/port/1
{"1": [{"rx_packets": 0, "tx_packets": 0, "rx_bytes": 0, "tx_bytes": 0, "rx_dropped": 1, "tx_dropped": 0, "rx_errors": 0, "tx_errors": 0, "rx_frame_err": 0, "tx_frame_err": 0, "rx_crc_err": 0, "tx_crc_err": 0, "collisions": 0, "duration_sec": 370, "duration_nsec": 32300000, "port_no": "LOCAL"}, {"rx_packets": 353677, "tx_packets": 120373, "rx_bytes": 19142644234, "tx_bytes": 7947278, "rx_dropped": 0, "tx_dropped": 0, "rx_errors": 0, "tx_errors": 0, "rx_frame_err": 0, "tx_frame_err": 0, "rx_over_err": 0, "tx_over_err": 0, "collisions": 0, "duration_sec": 370, "duration_nsec": 40900000, "port_no": 1}, {"rx_packets": 120373, "tx_packets": 353682, "rx_bytes": 7947278, "tx_bytes": 19142644564, "rx_dropped": 0, "tx_dropped": 0, "rx_errors": 0, "tx_errors": 0, "rx_frame_err": 0, "tx_frame_err": 0, "rx_over_err": 0, "tx_over_err": 0, "rx_crc_err": 0, "tx_crc_err": 0, "collisions": 0, "duration_sec": 370, "duration_nsec": 40900000, "port_no": 2}, {"rx_packets": 0, "tx_packets": 1, "rx_bytes": 0, "tx_bytes": 42, "rx_dropped": 0, "tx_dropped": 0, "rx_errors": 0, "tx_errors": 0, "rx_frame_err": 0, "tx_frame_err": 0, "rx_over_err": 0, "tx_over_err": 0, "rx_crc_err": 0, "tx_crc_err": 0, "collisions": 0, "duration_sec": 370, "duration_nsec": 40900000, "port_no": 3}]}mininet@mininet-vm:~$
mininet@mininet-vm:~$ curl http://localhost:8080/v1.0/topology/hosts
[{"mac": "02:76:d7:34:73:2d", "ip4": ["10.0.0.1"], "ip6": [], "port": {"dpid": "0000000000000001", "port_no": "00000001", "hw_addr": "4e:8f:ab:3c:25:8e", "name": "s1-eth1"}, {"mac": "66:aa:cc:a8:29:24", "ip4": ["10.0.0.2"], "ip6": [], "port": {"dpid": "0000000000000001", "port_no": "00000002", "hw_addr": "6a:21:30:47:fb:b1", "name": "s1-eth2"}}]mininet@mininet-vm:~$
```

Conclusion

This lab demonstrates the power of SDN by dynamically controlling traffic using OpenFlow meters and verifying controller behavior through REST APIs. The experiment highlights flexibility, programmability, and observability as key advantages of SDN.