SDN Lab Guide: Dynamic Traffic Throttling & REST API Verification

Q1. Dynamic Traffic Throttling using Mininet and Ryu

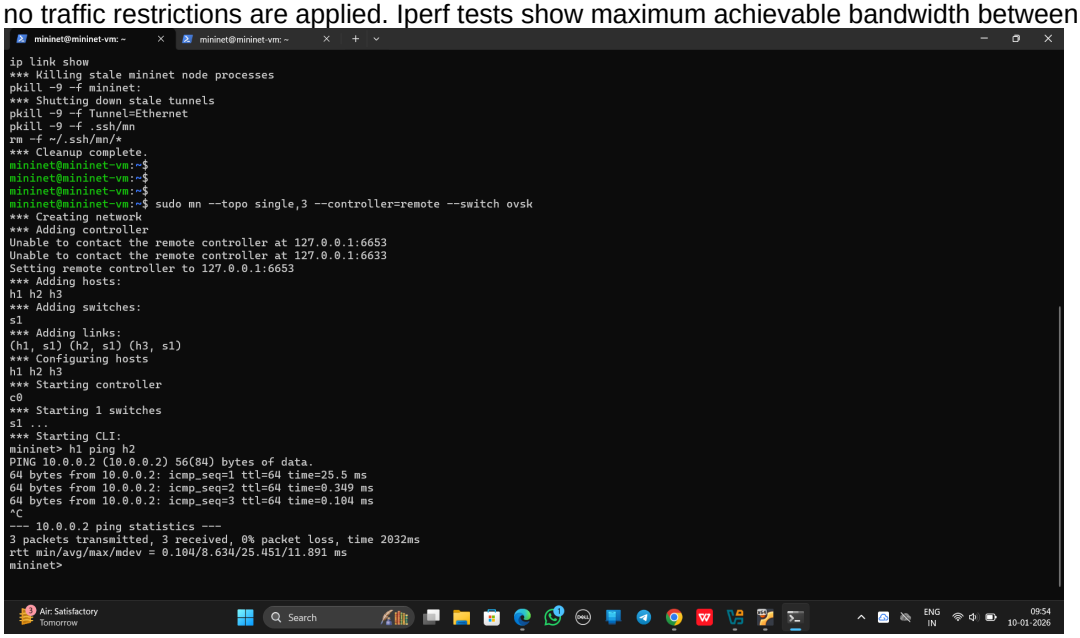
This experiment demonstrates how Software Defined Networking (SDN) enables dynamic traffic throttling using OpenFlow meters. A simple Mininet topology is controlled by a Ryu controller. Traffic from a selected host is rate-limited during peak usage without restarting the network.

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.

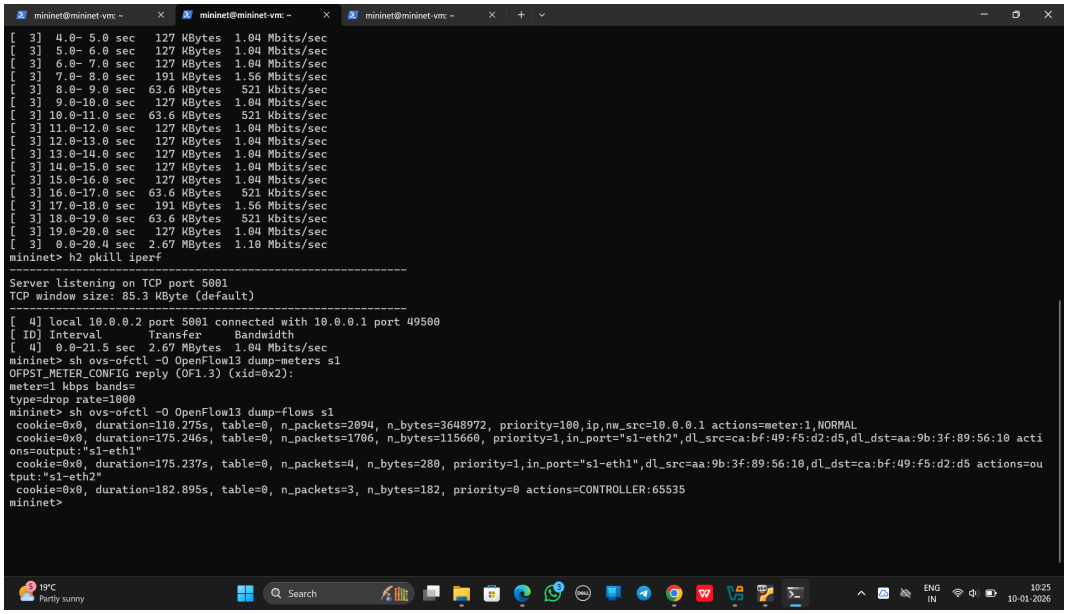
Normal Traffic Behavior

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



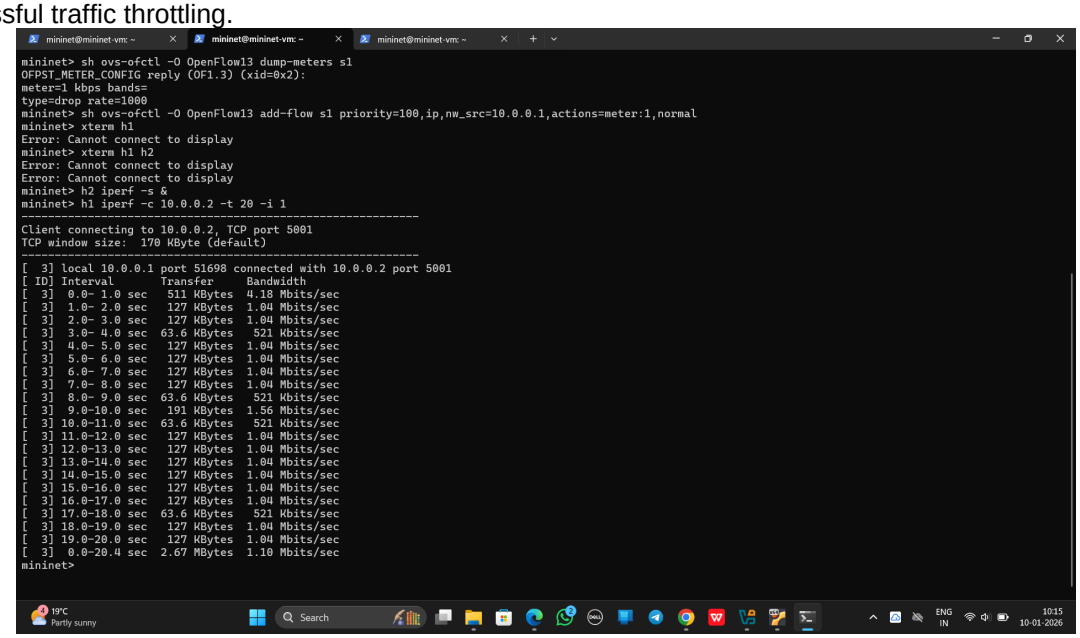
Applying Meter and Flow Rules

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.



Throttled Traffic Observation

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

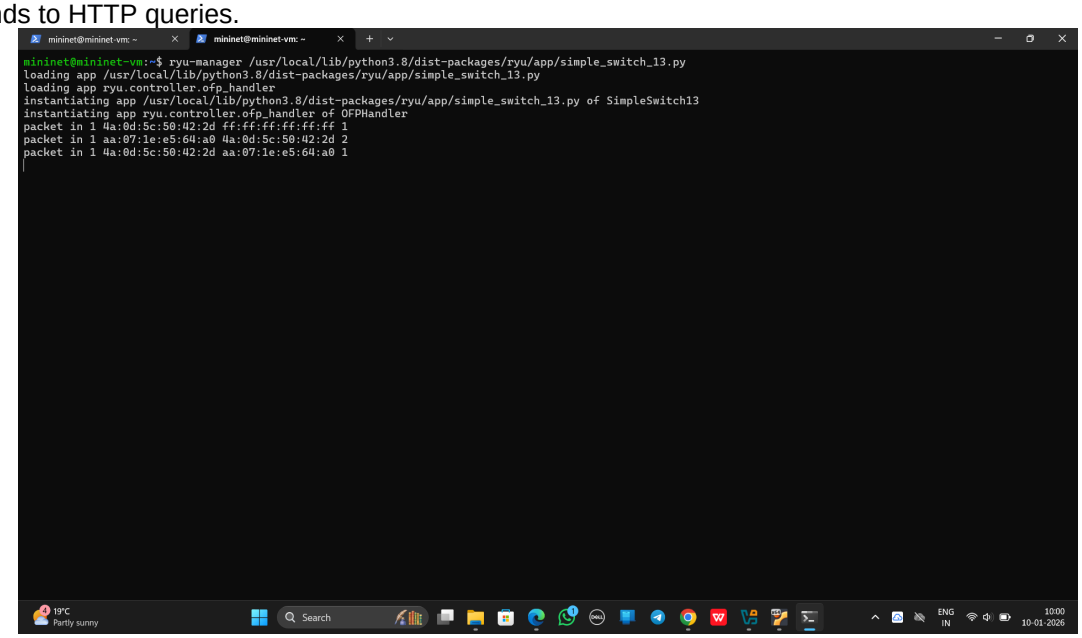


Q2. Verifying Controller Decisions using Ryu REST APIs

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.

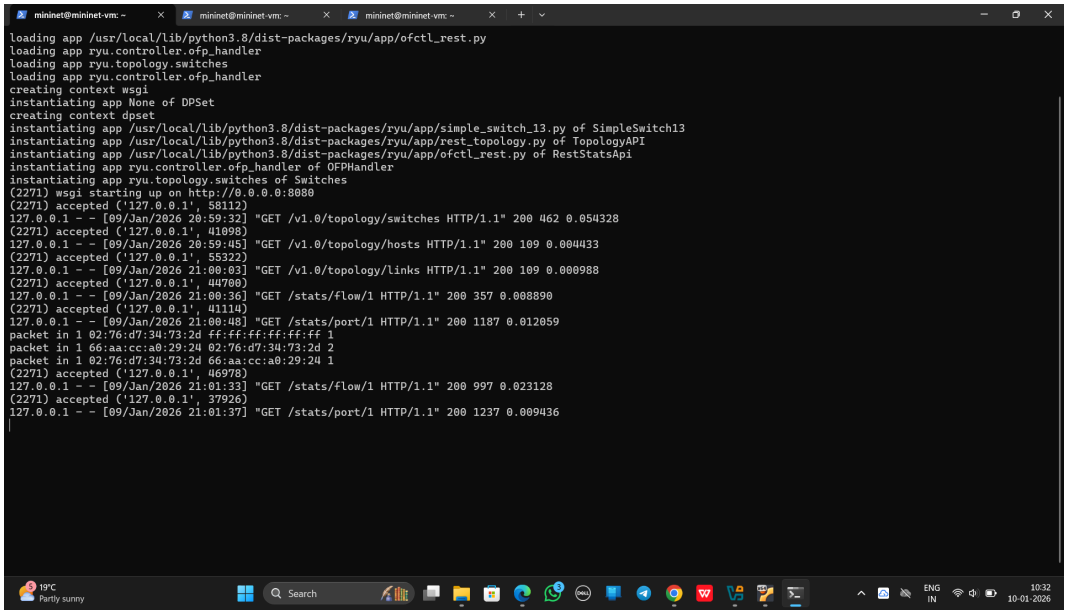
Controller Initialization

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



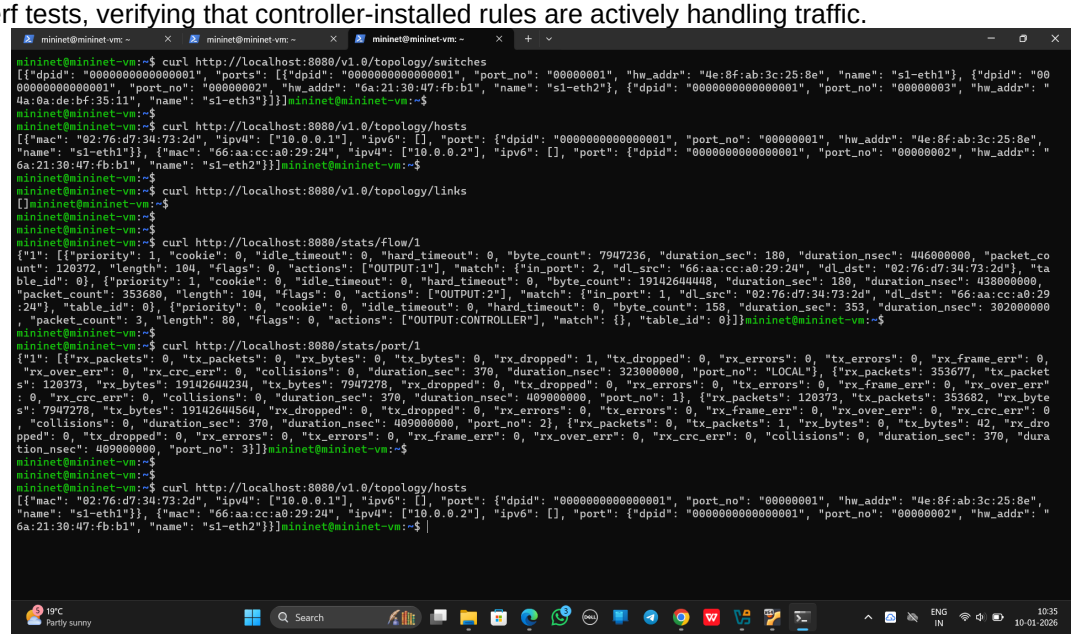
Topology Discovery via REST

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



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.



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.