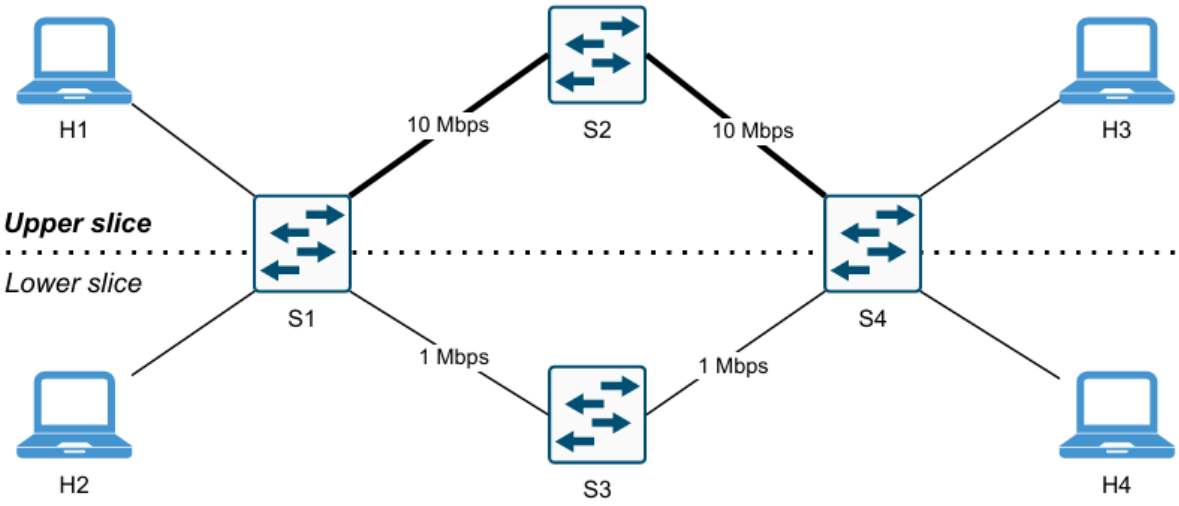
SDN Project Work: Network Slicing with Ryu and Mininet

1. Project Overview

The purpose of this project is to implement **network slicing** in an **SDN environment** using **Mininet** as the network emulator and **Ryu** as the SDN controller. The implementation will focus on:

1. **Topology Slicing** – Restricting communication paths between specific hosts.
2. **Service Slicing** – Prioritizing specific types of traffic (e.g., video) over others.

These concepts are commonly used to achieve traffic isolation, Quality of Service (QoS), and efficient network resource utilization.



2. Topology Slicing

Objective

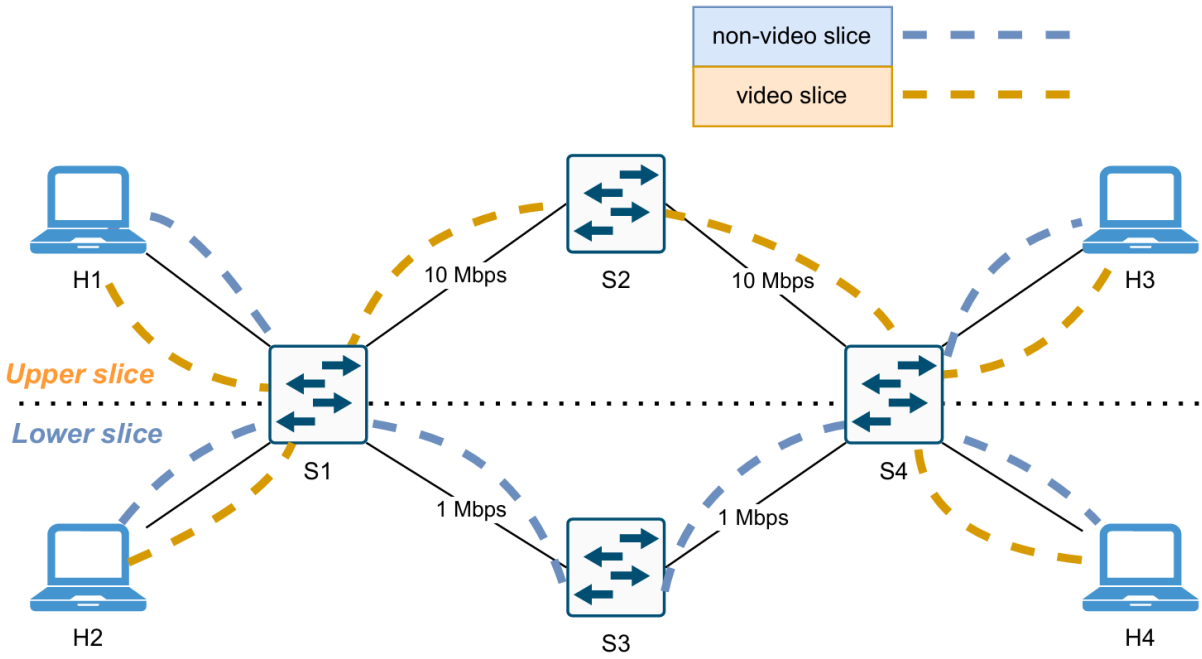
Implement a sliced network where specific hosts can only communicate through predefined paths. Specifically:

- **Host H1 should communicate only with H3** via an **upper path (slice)**.
- **Host H2 should communicate only with H4** via a **lower path (slice)**.

Steps to Implement and verify

1. **Define the network topology** in Mininet. NB: the topology in the figure is illustrative. Feel free to modify it, the important thing is that *at least* two different slices are present.

2. **Modify the Ryu SDN controller** to enforce topology slicing using flow rules based on MAC addresses (use mac_to_port dictionary) to allow the communication between H1 and H3 through the upper slice and the communication between H2 and H4 through the lower slice.
3. **Verify topology slicing** using Mininet's pingall command to ensure that 1) only H1-H3 and H2-H4 communications are possible, 2) the flow H1-H3 goes through the upper slice and the flow H2-H4 goes through the lower slice (e.g. through Wireshark or TCP dump)



3. Service Slicing

Objective

Implement **traffic-based slicing** where **video traffic** (identified as **UDP traffic on port 9999**) is prioritized over other types of traffic.

Steps to Implement

1. **Modify the Ryu controller** to identify traffic based on **destination port and transport protocol**.
2. **Set flow rules** in OpenFlow to classify and route video traffic separately from other traffic.
3. **Verify the slicing policy** using traffic generation tools like **iPerf**.

Testing

- Ensure that all hosts can **ping each other** (basic connectivity).

- Use **iPerf** to generate **UDP traffic on port 9999** and verify that it is routed through the dedicated slice.
- Compare throughput for **video traffic vs. other traffic**, confirming that video traffic has priority.

---

4. Possible Extensions

To further improve the slicing implementation, the following extensions can be considered:

1. **Dashboard for Visualization**
    - Develop a real-time monitoring dashboard to visualize network statistics (e.g., traffic flow bandwidth, delay.. for the different flows).
2. **Advanced Traffic Generation and Classification**
    - Use **D-ITG** (Distributed Internet Traffic Generator) to generate diverse traffic profiles (e.g., video and normal traffic).
    - Classify traffic based on **packet statistics** rather than just port numbers.
3. **Dynamic Slicing**
    - Allow **normal traffic to use the video slice** when bandwidth is available, but ensuring that video traffic always has priority.