Setup

- **Create an Ubuntu 22 VM and put it on a hypervisor at your choice (e.g. VMware or Virtualbox)**
- Install the required packages with the following commands:
    - sudo apt install vim
    - sudo apt install git
    - sudo apt install python3-pip
    - sudo pip install ryu
    - sudo apt install d-itg
    - sudo apt install nload

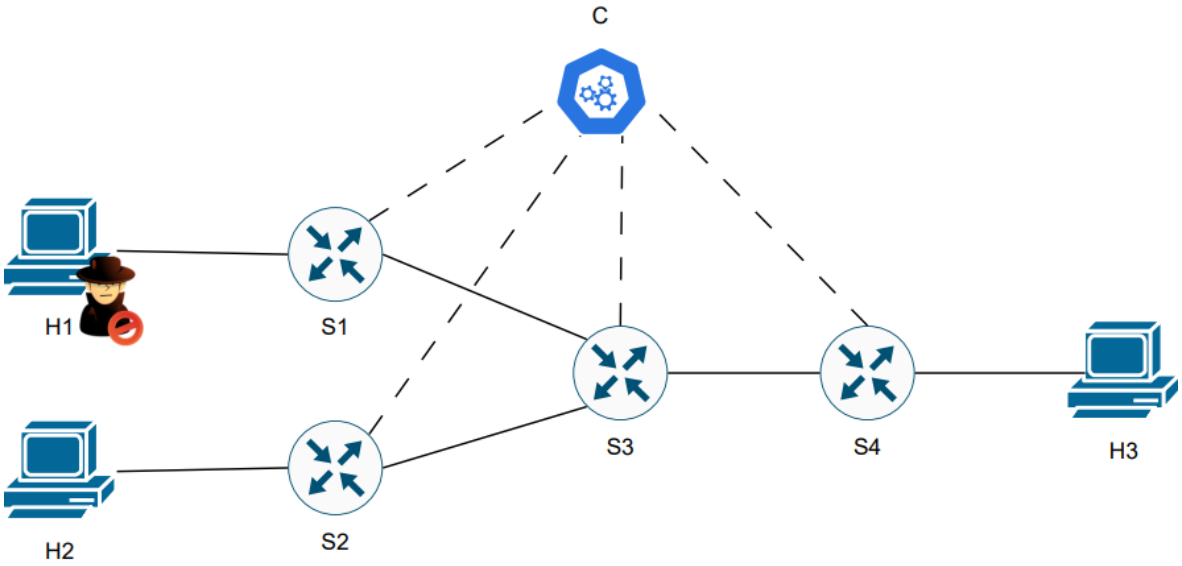
- **Install mininet with the following procedure**
    - git clone https://github.com/mininet/mininet
    - cd mininet
    - git tag
    - git checkout -b mininet-2.3.0 2.3.0
    - cd ..
    - sudo PYTHON=python3 mininet/util/install.sh -nv
    - sudo mn --switch ovsbr --test pingall # Test Mininet installation

SDN Project Work (Firewall)

The primary objective is twofold: firstly, to develop a robust monitoring system capable of detecting abnormal traffic patterns indicative of a Denial of Service (DoS) attack, and secondly, to implement an automated mitigation mechanism to safeguard network operation for legitimate users.

Topology definition, traffic generation, performance evaluation

a. Define the following topology in Mininet through Python code, using Open vSwitch:



b. Guarantee basic reachability through the controller. (you can use for example simple_switch13.py)

c. Generate traffic:

H1 is controlled by an attacker that generates a large amount of traffic towards H3 causing congestion on the links, while H2 generates normal traffic towards H3.

- For traffic generation you can use iperf or D-ITG.
- H1 generates UDP traffic, while H2 generates TCP traffic, both have constant bitrate.

d. Report performance degradation for H2 owing to H1 DoS attack

E.g. *show packet loss or increased latency or throughput degradation.*

Design and implement a mitigation/remediation for the attack

a. Traffic Monitoring:

Implement logic within the Ryu app to continuously monitor the traffic on switch ports. Use `EventOFPPortStatsRequest` and `EventOFPPortStatsReply` to collect port statistics periodically.

- Retrieve statistics for each port of every switch in the network.
- Calculate throughput (e.g., bytes received per second) for each port. *

- Analyze this data to determine if any port's throughput exceeds the defined threshold.
- If it is the latter case raise an alarm.
- *Show the alarm variable changing value when the throughput exceeds the defined threshold.*

*- notice that collected metrics with StatsRequest and StatsReply methods (rx_packets, tx_packets, rx_bytes, tx_bytes …) are cumulative: design a method to consider their value each time interval
(e.g. keep the previous value and compute (new-previous)/interval)

b. Remediation

When a port's throughput surpasses the threshold:

- Identify the switch and port that is receiving high traffic and block it.
    - Using OpenFlow rules (`OFPFlowMod`), instruct the switches to block traffic from the identified port/ports by installing flow entries that drop packets from those port/ports. (Pay attention to the flow priority!) Push the new flow entries to the switches experiencing high traffic.

3. Integration & Execution:

Combine the Mininet topology setup and the Ryu controller application.

- Start the Mininet network with the defined topology and start the normal Ryu Controller application without the monitoring and remediation mechanism designed.
- Generate traffic.
- Report what happens.
- Run the Ryu controller application with traffic monitoring and traffic-blocking logic when thresholds are exceeded.
- Generate traffic.
- Report what happens by comparing the two situations in terms of the relevant metrics.

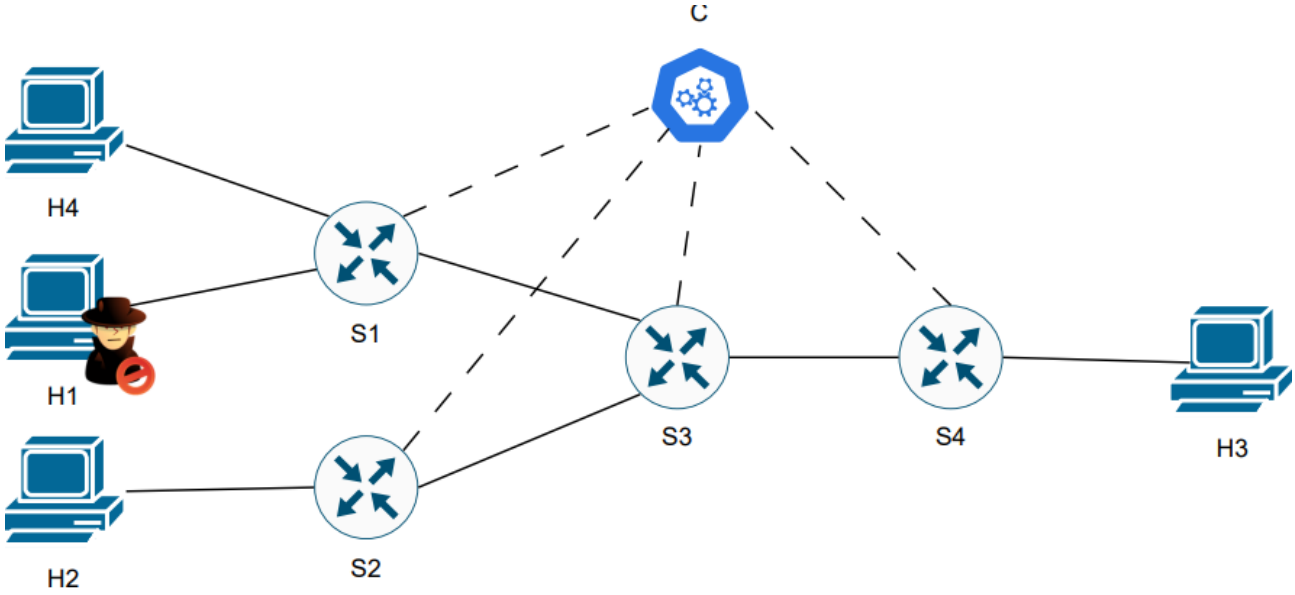
Additional Considerations:

- **Threshold Determination**:** Define a reasonable threshold for throughput based on network capacity, desired performance, and potential congestion levels. (e.g. set it considering the network links bandwidth)

- **Multithreading**:** You can use multithread programming (e.g. launch a thread for monitoring in the app class constructor)

Optional features:

1. H1 generates variable bitrate traffic: make the remediation mechanism dynamic: (e.g. after some time that the throughput does not exceed the threshold, unblock the port)



2. Set rules to minimize the impact on the legitimate hosts: e.g. consider the updated topology: a new host (H4) is attached to S1 and generates traffic to H3. What happens if you block all the traffic from the port connecting S3 to S1? (… the piracy shield case)
3. Any other additional features from you are welcome. Be creative!

Output of the project:

- Python code for the topology
- Phyton code for the Ryu controller with monitoring and remediation actions
- Detailed documentation
    - Describe how you solved every challenge and every design choice you made: e.g. link bandwidth, threshold selection, time interval for monitoring…
    - Not a single block of text! Image for the topology, screenshots of the execution of the experiments (e.g. output of the ping, iperf commands, or d-itg logs), plots of latency/throughput/jitter values…

Useful links:

OpenFlow v1.3 Messages and Structures — Ryu 4.34 documentation

Traffic Monitor — Ryubook 1.0 documentation (osrg.github.io)

…