Computer Networks, Spring 2026

Instructor: Shashi Prabh

Project: Software-Defined Networking (SDN) with Mininet

1 Introduction

In this project, you will move beyond traditional hardware-based networking and explore Software-Defined Networking (SDN). You will use the Mininet network emulator to create virtual topologies and control them using an OpenFlow-capable controller. *This project is mandatory and to be done individually on your own laptops.*

2 Background: Software-Defined Networking

2.1 The Traditional Network Architecture Problem

In conventional networks, data forwarding and control decisions are tightly integrated within the same hardware devices (routers and switches). Each device independently makes forwarding decisions based on its own routing tables and policies. This creates several challenges:

- **Limited flexibility**: Changing network behavior requires modifying individual devices and their configurations.
- **Vendor lock-in**: Network functionality is tied to proprietary hardware implementations.
- **Difficulty in rapid innovation**: New network services and policies are cumbersome to deploy.
- **Poor network visibility**: Lack of a centralized view makes network troubleshooting and optimization difficult.

2.2 The SDN Paradigm

Software-Defined Networking addresses these challenges by *decoupling the control plane from the data plane*:

- **Data Plane**: Consists of simple, "dumb" forwarding devices (switches) that execute packet forwarding rules. These switches contain flow tables that specify how to handle packets matching certain criteria.
- **Control Plane**: Runs on a centralized, external controller that computes network policies and dynamically installs forwarding rules on switches. The controller has a global view of the entire network.

This separation enables network administrators to program network behavior much like writing software—defining policies in a controller program rather than configuring individual devices. The controller communicates with switches using a standardized protocol, most commonly *OpenFlow*.

2.3 OpenFlow Protocol

OpenFlow is a communication protocol that allows a controller to interact with network switches. Key components include:

- **Match fields**: Headers (source/destination IP, port numbers, MAC addresses, etc.) used to identify packets.
- **Actions**: Instructions for matched packets (forward to port, drop, modify header, etc.).
- **Flow table**: A table on each switch storing match-action rules. When a packet arrives, the switch checks it against all entries in the flow table and executes the corresponding action.

When a switch receives a packet that does not match any flow table entry, it typically forwards the packet to the controller, which decides how to handle it and installs new flow rules for future packets. This dynamic control allows for flexible and programmable network behavior, enabling applications such as traffic engineering, load balancing, and security policies.

3 Background: Mininet

3.1 Introduction to Mininet

Mininet is a network emulator that allows you to create realistic virtual network topologies on a single machine. Instead of purchasing expensive hardware or setting up multiple physical machines, you can:

- Quickly prototype and test network architectures.
- Experiment with different topologies and configurations.
- Develop and debug SDN controllers without complex setup.
- Run standard Linux networking tools and commands within virtual hosts.

3.2 Key Concepts in Mininet

- **Hosts**: Virtual machines that act as end systems (e.g., computers on a network). You can run applications on them and send/receive traffic.
- **Switches**: Virtual OpenFlow-capable switches that forward traffic between hosts. They connect to a controller to receive flow rules.
- **Links**: Virtual connections between hosts and switches or between switches, with configurable bandwidth, latency, and loss.
- **Controllers**: SDN controllers (such as Ryu, POX, or others) that manage switch behavior by installing flow rules via OpenFlow.
- **Topologies**: Network configurations defined by the arrangement and connectivity of hosts, switches, and links.

3.3 How Mininet Works

Mininet uses Linux containers (namespaces) and virtual network interfaces to create a network topology on a single physical machine. When you run a Mininet topology:

1. A Python script defines the network structure (hosts, switches, links).
2. Mininet creates isolated Linux namespaces for each host.
3. Virtual network devices (switches) are instantiated using Open vSwitch (OVS), which implements the OpenFlow protocol.
4. An SDN controller connects to the switches and controls their behavior.
5. You can interact with hosts using standard Linux commands (e.g., `ping`, `iperf`) to test connectivity and measure traffic.

The documentation for Mininet is available at `http://mininet.org/`, which provides detailed instructions and examples.

3.4 Controller Behavior

In this project, you will work with switches operating in different control modes:

- **Learning (default)**: The switch learns MAC addresses dynamically. When it receives a packet from an unknown source, it floods the packet to all ports. As it observes replies, it learns the mapping of MAC addresses to ports and installs corresponding flow rules.
- **Controller-based**: The switch relies on the controller for all forwarding decisions. When a packet arrives that doesn't match any flow rule, the switch forwards it to the controller, which decides the action.
- **Hub behavior**: Every incoming packet is flooded to all ports (except the source), simulating the behavior of a network hub.

4 Setup

1. Install Mininet on your machine. We recommend using a Virtual Machine (VM) if you are not running Linux natively. Instructions: `http://mininet.org/download/`
2. Install an SDN controller (e.g., Ryu or POX).

5 Tasks

5.1 Custom Topologies

Create Python scripts to experiment with the following custom topologies in Mininet:

1. A single switch connected to 4 hosts.
2. A linear topology with 3 switches, each with one host.

5.2 Controller Experimentation

1. Run Mininet with the default "Learner" switch behavior. Observe how the switch populates its flow table as hosts ping each other.
2. Implement or modify a controller script to change the switch behavior:
    - **Hub Behavior**: Every incoming packet is flooded to all ports (except the source).
    - **Static Firewall**: Configure the controller to drop all traffic between Host 1 and Host 3, while allowing all other traffic.

6 Submission

Submit your Python topology scripts, your controller code, and a report (with few screenshots) showing the flow table entries (`ovs-ofctl dump-flows`) for your static firewall implementation.

7 Evaluation Criteria

☐ Successfully installed and verified Mininet environment. TA: _____

☐ Can create custom topologies using Python scripts. TA: _____

☐ Demonstrated a functional SDN firewall or custom flow logic. TA: _____