CS416 Project 1: Implementation of Ethernet Switching

Learning objectives:

1. Describe the functionality and operation of Ethernet switches.
2. Implement the Ethernet Learning Switch algorithm.
3. Construct a local-area network by implementing virtual switches and hosts.

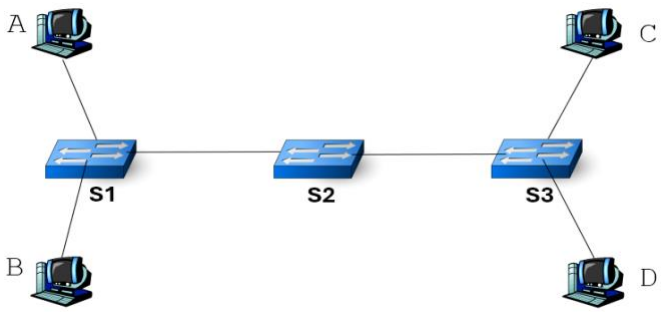
What to submit:

1. The java source files.
2. The configuration file.
3. A note summarizing:
    a. Each member's attendance at group meetings, including in-class collaboration sessions and any additional meetings scheduled by the group.
    b. Each member's contribution to the project as agreed upon by all members.

(Each group only needs to submit one copy on Canvas. Please include a submission comment listing the names of all group members.)

Instructions:

1. Provide a Java implementation of a virtual Ethernet switch.
    a. Implement the Ethernet Learning Switch algorithm to build the switch table and forward frames.
2. Provide a Java implementation of a virtual host.
    a. Your host implementation must be capable of both sending and receiving virtual frames.
3. Provide the configuration file that encodes the following local-area network (LAN) topology (this topology will be used in the demo):



    a. The config file should include the *IP address*, *port number*, and *ID* (e.g., S1, A) of each device, and how they are connected.
    b. The config file should also specify how the devices are connected, i.e., it should include all the links.

4. When a switch or host starts, it gets its ID (e.g., S1, A) as a command-line argument, specified through intelliJ. For simplicity, **a host shall use its assigned ID as its *virtual* MAC address**.
5. When a switch or a host starts, it parses the config file and finds out its neighbors. For example, switch S1 in the example topology should find three neighbors: host A, host B, and switch S2. It sees the IP address and port number of each neighbor from the config file.
6. Each switch or host has an IP address and a UDP port number, which together serve as the "physical" address of the device.
7. A switch has multiple *virtual* ports, and **each virtual port should be "named" using the IP address *and* the port number of the neighbor switch or host**.
    a. As an example, the left virtual port of S2 should be named using S1's IP address and S1's port number.
8. For debugging purposes, **the switch must print out the entire switch table every time a new entry is added**.
9. A host is always waiting for the user to enter a short message together with the intended receiver. The host then generates a virtual frame.

    A virtual frame has three parts:
    a. the virtual source MAC address
    b. the virtual destination MAC address (as entered by the user)
    c. the short message (as entered by the user)

    As an example, a virtual frame from A to D with the message "hello" will look like the following:

    "A:D:hello"
10. The host then creates a UDP packet to carry the virtual frame and **sends it to the connected switch** using the switch's IP address and port number.
    a. Important: **Do *not* send the UDP packet directly to the destination host**. All communication must flow through the virtual network to emulate real Ethernet switching, i.e., the UDP packet should only be sent to the IP and port of the immediately connected neighbor as defined in the topology.
11. In a separate thread, a host is always waiting to receive virtual frames from the connected switch. When a frame arrives, **the host must print the message and the source host ID**. Additionally, it should check if the virtual destination MAC address matches its own; **if the two do not match, the host must print a debug message to indicate a MAC address mismatch** (this must be a flooded frame).
12. To run the example network topology, you will need to manually start three separate instances of the virtual switch and four separate instances of the virtual host.

a. To run multiple instances of the same program in IntelliJ, you will need to enable the “Allow multiple instances” option. This is located in “Modify Run Configuration” and then “Modify Options”.
b. During the demo session, your group must distribute these instances across all available laptops. Make sure your config file uses the actual IP addresses for the instances so they can communicate over the network.

Grading:

This project is worth a total of 100 points, distributed as follows:

- 50 points for the virtual switch implementation.
- 30 points for the virtual host implementation.
- 20 points for participation: to earn these points, you must:
    - Attend all collaboration sessions, both in class and as scheduled by your group, and the demo session. (10 points)
    - Actively contribute to the project in a meaningful way. (10 points)

Partial credit: If your implementation is only partially correct, you will receive partial credit proportional to the level of functionality achieved.

Demo session: you will be required to give a live demonstration of your work, which will be the primary basis for grading. Details of the demo session will be provided separately.