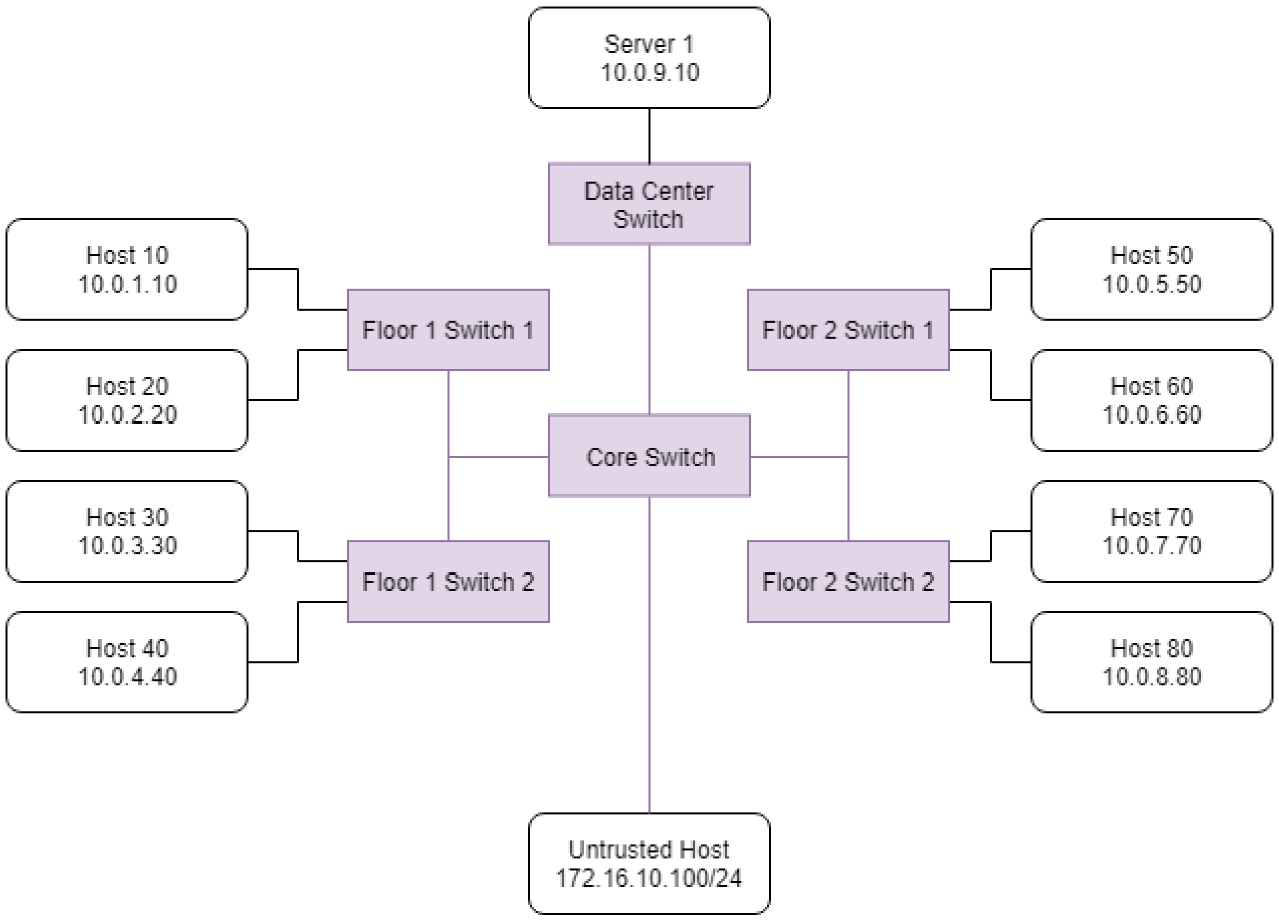
Final Project: Implementing a Simple Router

In the previous lab you implemented a simple firewall that allowed ICMP and ARP packets, but blocked all other packets. For your final project, you will be expanding on this to implement routing between subnets, and implementing firewalls for certain subnets. The idea is to simulate an actual production network.

Assignment:

For this lab, we will be constructing a network for a small company. The company has a two-floor building, with each floor having its two switches with their own subnets. Additionally, we have a switch and subnet for all the servers in the data center, and a core switch connecting everything together.

The topology will look as follows:



Your goal will be to allow traffic to be transmitted between all the hosts. In this assignment, you will be allowed to flood all non-IP traffic in the same method that you did in Lab 3 (using a destination port of of.OFPP_FLOOD). However, you will need to specify specific ports for all IP traffic. You may choose your preferred method to accomplish this -- however, you may find it easiest to determine the correct destination port by using the destination IP address and source IP address, as well as the source port on the switch that the packet originated from.

Additionally, to protect the company servers from the untrusted Internet, you will be blocking all IP traffic from the Untrusted Host to Server 1. To block the Internet from discovering our internal IP addresses, we will also block all ICMP traffic from the Untrusted Host. You will need to explain how you implemented the various requirements and show that they work properly.

Provided Code:

Available in a ZIP file here.

We have provided you with starter code (skeleton files) to get you started on this assignment. The controller file (final_controller_skel.py) needs to be placed in ~/pox/pox/misc, and the mininet file (final.py) should be placed in your home directory (~). This time, you will need to modify both files to meet the lab requirements.

You will be using slightly different commands to create the Hosts and Links in the Mininet file to give you more information to make decisions within the Controller file. Additionally, you will notice that you have additional information provided in the do_final function. This is documented in the comments within the files.

Summary of Goals:

- Create a Mininet Topology (See Lab 1 for help) to represent the above topology.
- Create a Pox controller (See Lab 3 for help) with the following features:
    - All hosts able to communicate, EXCEPT:
        - Untrusted Host cannot send ICMP traffic to Host 10, Host 20, Host 30, Host 40, Host 50, Host 60, Host 70, Host 80 or Server 1.
        - Untrusted Host cannot send any IP traffic to Server 1.

Project Report:

You will have to write a report of the work done in this project. It should have two main sections, in the first one you will present the logic of your controller in plain English; in the second part you will present test evidence to show that your code works, or explain any problems that might have happened.
You may test with any mininet commands such as ping and iperf or observing packets with Wireshark inside your VM. Please include screenshots of the executed tests.

Grading Rubric:

Total: 100 points

30 points: Mininet Topology
    10: Devices successfully created.
    10: Links successfully created.
    10: IP addresses correct.
50 points: Pox Controller
    25: All hosts can communicate.
        -15 if rules not installed in flow table.
        -20 if IP traffic is implemented using OFPP_FLOOD.
    15: Untrusted Host cannot send ICMP traffic to Hosts 10-80

-10 if Untrusted Host cannot send ANY traffic to these hosts

10: Untrusted Host cannot send any traffic to Server 1

20 points: Project Report

5: Screenshot showing the result of running pingall

5: Screenshots showing the result of running iperf between two company hosts, between one company host and untrusted host, and between untrusted host and server

5: Screenshot of running `dpctl dump-flows` after running the above commands

5: High-level explanation of the logic of the controller

Deliverables:

1. **final.pdf**: Project report, as described above.
2. **final.py**: topology file that you created from the skeleton.
3. **final_controller.py**: controller file that you created from the skeleton.